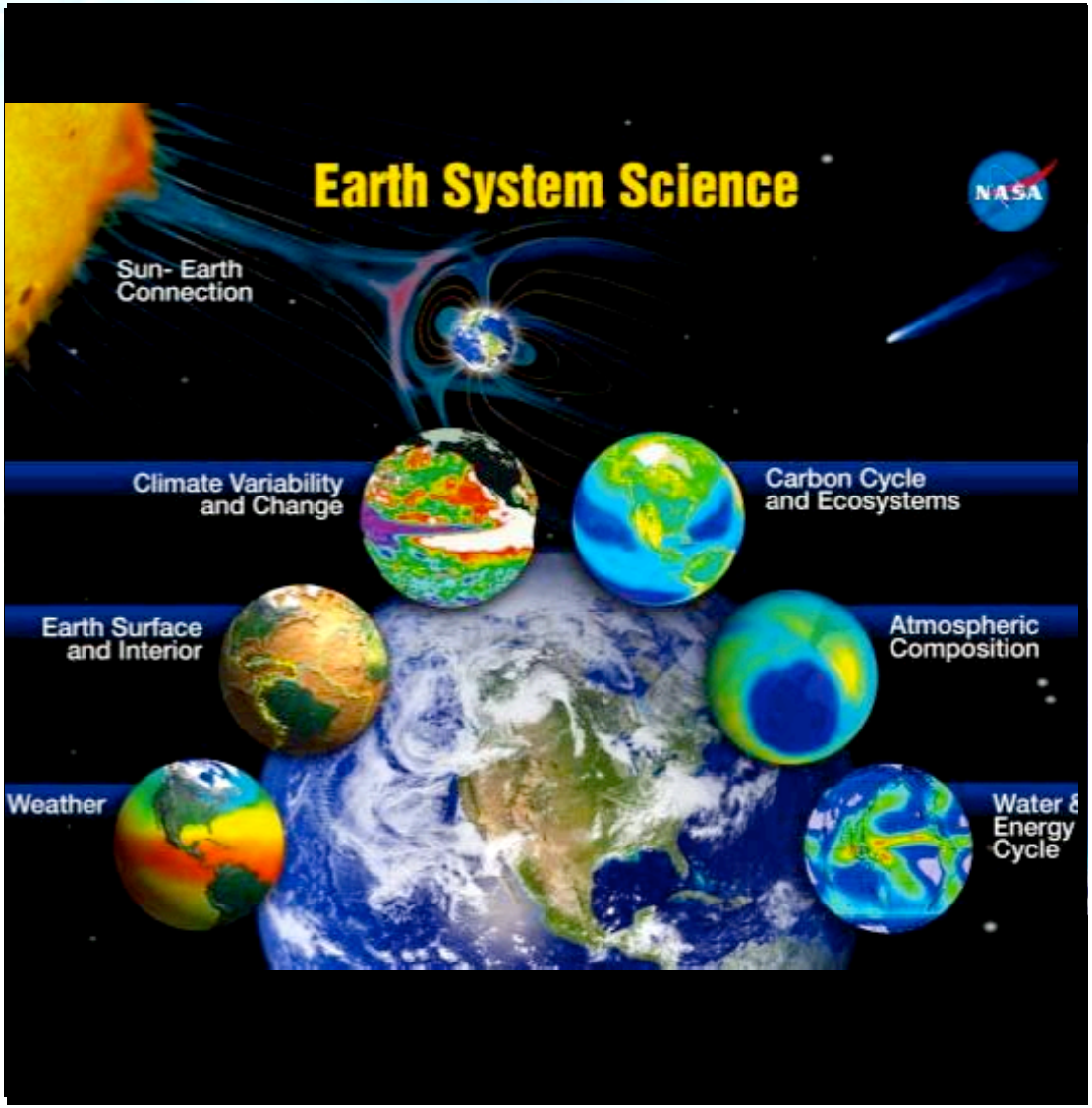
SciFlo™: Scientific Knowledge Creation on the Grid Using a Semantically-Enabled Dataflow Execution Environment

GENESIS ESIP:
Brian Wilson, Tom Yunck, Elaine Dobinson, Benyang Tang, Gerald Manipon, Dominic Mazzoni, Amy Braverman, and Eric Fetzer
Jet Propulsion Laboratory

Do multi-instrument science by *authoring* a dataflow doc. for a *reusable* operator tree. Access scientific data by *naming* it.

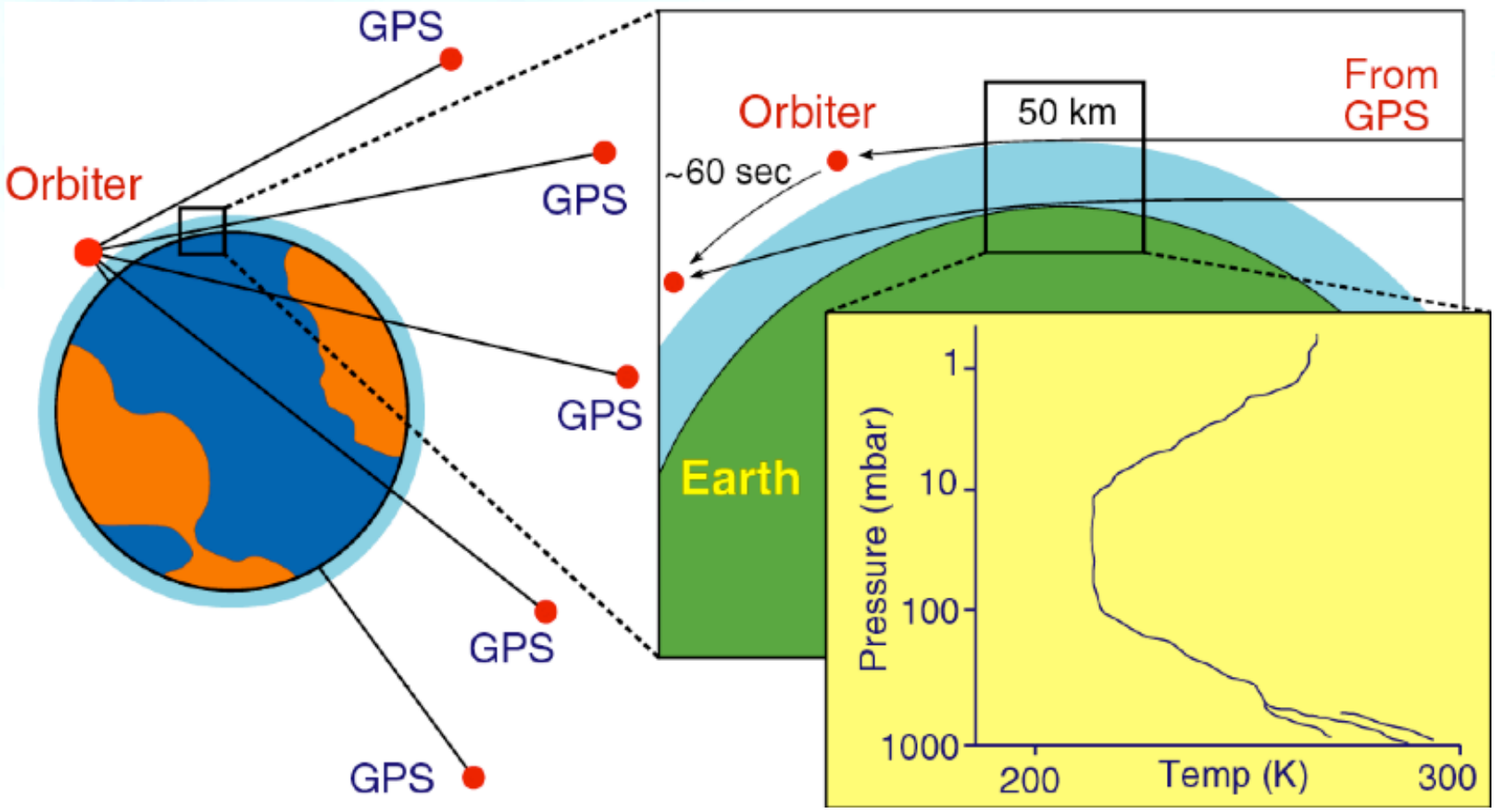
SciFlo: Scientific Knowledge Flow

GENESIS Project: Multi-Instrument Atmospheric Science



- **Characterize**
  Earth's varied behavior
- **Understand**
  the Earth as an integrated system
- **Predict**
  Earth's response to complex forcings

Atmospheric Temperature Profiling with GPS Limb-Sounding



Goal: Large-scale Science

JPL

- Measurement-Based Atmospheric Science (Data Fusion):
  - Combine years (terabytes) of atmospheric data from multiple EOS instruments – AIRS, MODIS, MISR, and GPS – to produce calibrated, long-term climate records (means, variability, correlations) of atmospheric temperature, water vapor, cloud properties, aerosols, etc.
  - Publish merged, co-registered, multi-instrument datasets
- Problems:
  - Huge data volume, located in multiple archives, varied formats
  - Lack of verifiable, reusable scientific data fusion algorithms
- Potential IT Solution: Grid Workflow
  - Distributed data processing via dataflow engine
  - Leverage Web Services standards, open source
  - But can it be lightweight? Globus toolkit is not.

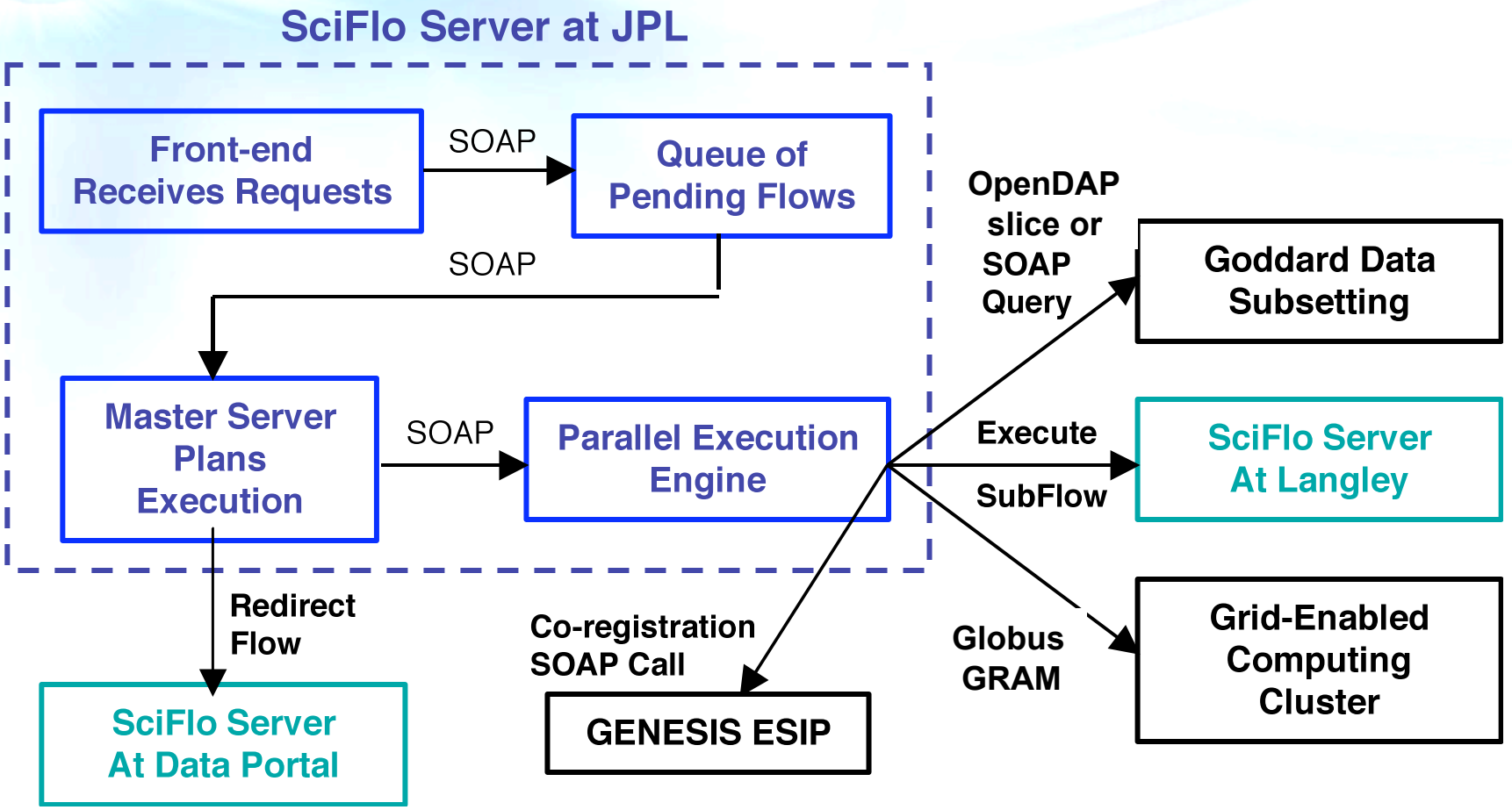
"The Grid" Concept

"On-Demand"
Virtualization

GENESIS SciFlo Engine

- "SciFlo" stands for Scientific Dataflow
- Automate large-scale, multi-instrument science processing by *authoring* a dataflow document that specifies a *tree* of web services & executable operators.
  - iEarth Visual Authoring Tool
  - Distributed Dataflow Engine choreographs execution.
  - Move operators (executables) to the data.
  - Built-in reusable operators provided for many tasks such as subsetting, co-registration, regridding, data fusion, etc.
  - Custom operators easily plugged in by scientists.
  - Leverage convergence of Web Services (SOAP) with Grid Services (Globus toolkit v4).
- Hierarchical namespace of objects, types, & operators.
  - sciflo.data.EOS.AIRS.L2.atmosphericParameters
  - sciflo.operator.EOS.coregistration.PointToSwath

Distributed Computing Using SciFlo



Inject data query or flow execution request into SciFlo network from any node.

Outline

- Enabling Technologies
  - Web Services: XML, SOAP, WSDL
  - Grid Services: WS-ResourceFramework & Globus toolkit v4
  - Parallel dataflow engines
  - Semantic Web: OWL inference using metadata & ontologies
- SciFlo Distributed Dataflow System
  - Loosely-coupled distributed computing using Web (SOAP) and Grid services
  - Specifying a processing stream as an XML document
  - Dataflow engine for automated execution and load balancing
- Multi-Instrument Earth Science
  - Motivating Example: Compare the temperature & water vapor profiles retrieved from AIRS (Atmospheric Infrared Sounder) swaths and GPS limb soundings.

SciFlo's Innovation Themes

- Think in XML, not Java
- Metadata in XML, types in XML schema, semantics in OWL/RDF
- Publish algorithms as Web Services (software reuse)
- Declare distributed dataflow in a simple XML document
  - SciFlo engine automates Grid workflow execution
- Author new, custom services by writing a SciFlo doc. (no code)
  - Optional visual programming (layout) GUI
- Personal science notebook to store/query results
- Virtual datasets on demand from Grid workflows
- Peer-to-peer network of personal SciFlo nodes
  - Many "portals" serve multi-instrument, value-added datasets
- Capture semantic labels during workflow execution
- Discover new datasets/services using "smart" search
- Hybrid xml/binary object serialization
  - HDF & netCDF compatible

What are Web Services?

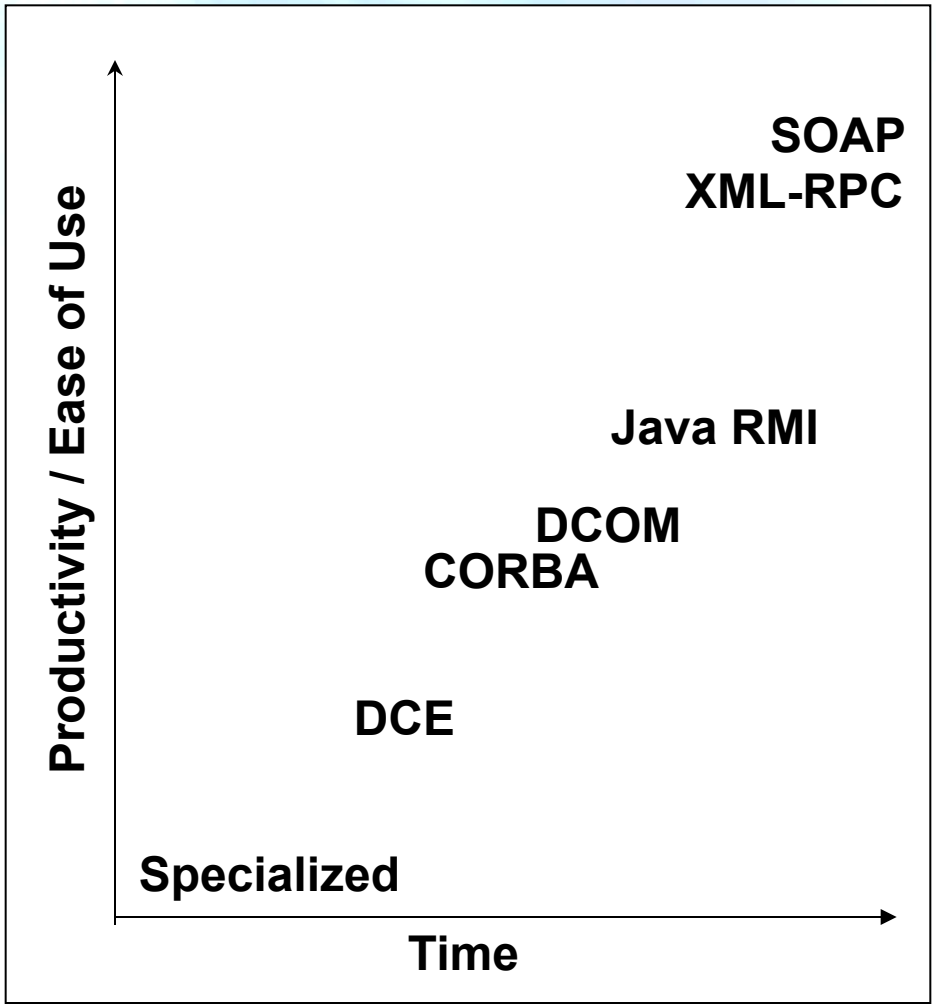
- web services (little w):
  - Any service available in a web browser
- Web Services (with a capital W):
  - Technical term
  - Service interface is fully structured in XML
  - Communications often done using Simple Object Access Protocol (SOAP)

Three Generations of the Web

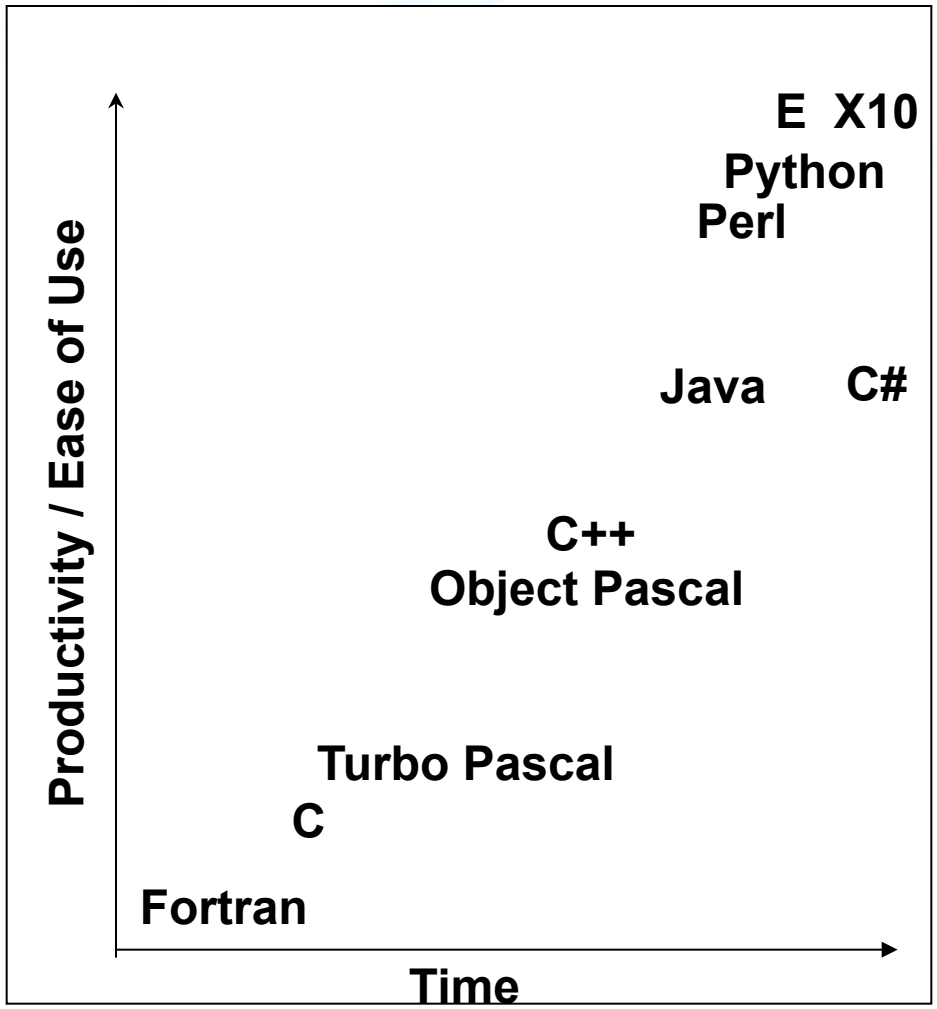
- 1st: Static HTML pages with Pictures!
  - Hyperlinks: click and jump!
  - Easy authoring of text with graphics (HTML layout).
  - Killer App: Having your own home page is hip.
  - Cons: Too static, One-way communication.
- 2nd: Dynamic HTML with streaming audio & video
  - Browser as an all-purpose, ubiquitous user interface.
  - Fancy clients using embedded Java applets.
  - Killer App: Fill out your time card on-line.
  - Cons: Applets clunky, ease of authoring disappears, information is still HTML (semi-structured).
- 3rd: SOAP-based Web Computing & Semantic Web
  - Exchange structured data in XML format (no fragile HTML); semantics ("meaning") kept with data.
  - Programmatic interfaces rather than just GUI for a human.
  - Killer Apps: Grid Computing, automated data processing.

SciFlo: Scientific Knowledge Flow

History of Distributed Computing



History of Programming Languages



Sad History of Function Call Interfaces

- Fortran – partially typed
  - Positional arguments only, by address, partially typed
- C/Pascal – fully typed
  - Added varargs for unknown number of arguments
  - But used pointers to return multiple results
- C++
  - Added function/operator overloading
- Java
  - Pointers hidden (yes!), but lost operator overloading
- Perl, Python, IDL – func(arg1, arg2, *array, **hash)
  - Added implicit array to catch variable number of args
  - Added implicit hash to catch named args
- XML – input & output are hierarchical XML documents
  - Flexible interface since can ignore unneeded tags
  - Hierarchy & dynamic typing imply rich interface
  - Use XQuery/XPath to extract/update needed tags

Web 2.0: Pervasive Use of XML

- Metadata in XML and data types in XML schema
- Distributed computing via XML messaging (SOAP)
- Service interfaces described in XML (WSDL)
- Services published in queryable catalogs (UDDI)
- Operators and data typed using XML schema and namespaces
- Semantic "kind" or meaning described using science ontologies (RDF/OWL)

Web Services "Stack"

- Metadata: XML and XML schema
- Messaging: Simple Object Access Protocol (SOAP)
- Service interfaces: Web Service Description Language (WDSL) and OWL-S
- Service registries: Universal Description, Discovery and Integration (UDDI) or P2P discovery
- Semantic Web: Ontologies or "concept maps" in Resource Description Framework (RDF) and Ontology Web Language (OWL)
- Grid Workflow: Many dataflow engines
  - SciFlo – Earth Science app, GENESIS ESIP
  - Kepler, Triana, Pegasus, etc.
  - Business Process Execution Language (BPEL) - IBM

What is Simple Object Access Protocol (SOAP)?

- Distributed Computing by Exchange of XML Messages
  - Lightweight, Loosely-Coupled API
  - Programming language independent (unlike Java RMI)
  - Transport protocol independent
- Multiple Transport Protocols Possible
  - HTTP or HTTPS (HTTP using SSL encryption) POST
  - Email (SMTP), Instant Messaging (MQSeries, Jabber)
  - Store & Forward Reliable Messaging Services (Java JMS)
  - Even Peer-to-Peer (P2P) protocols or LambdaRail
- SOAP Toolkits for all languages
  - Apache Axis for Java, modules for python/perl, Visual C# .Net.
- Web Service Description Language (WSDL)
  - Generate call to a service automatically from WSDL document.
  - Data types and formats expressed in XML schema.
  - Publish services in UDDI catalogs for automated discovery.

SOAP Software Toolkits

- Java (open source)
  - Install Java 1.4, Tomcat 4.1, and Apache Axis 1.0
  - Optionally install WSDL4J, UDDI4J, & WSIF from IBM
  - Configure Tomcat & Axis: classpaths, etc.
- Perl (open source)
  - Install perl 5.6 & SOAP::Lite module (contains UDDI::Lite)
- Python (open source)
  - Install python 2.2-2.4 and SOAP.py
  - Optionally install UDDI4py from IBM
- DotNet (not free, but cheap)
    - Install Visual Studio .Net with C# or VB compilers
    - Send a payment to MS
- C++ (open source)
  - Install gSOAP 2.7

Python Client Example

```
#!/usr/local/bin/python
# soap_sql.py

from SOAPpy import SOAP
#import SOAP

db = sys.stdin.readline()
lines = sys.stdin.readlines()
sql = string.join(lines, '\n')

server = SOAP.SOAPProxy(
   "https://genesis:passwd@gilaan.usc.edu:80/axis/GenesisSQL.wsdl",
   namespace = "urn:Genesis_SQL",
   encoding = None)

print server.executeQuery(SOAP.stringType(sql), SOAP.stringType(db))
```

Input:
jplsacc
select * from L2data where datasetid == '20010801_1350sac_g43_1p0'
and datatype == ‘gpsProfile’

Benefits of Using Web Services

- SOAP Messaging is programming language independent
  - Think in XML, not Java or C#.
- SOAP is decoupled from the transport protocol
  - Use Reliable Messaging, email, p2p, not just http
  - Can be asynchronous but guaranteed (with transactions)
- Software Reuse
  - Algorithms exposed as services are reusable!
  - Algorithms callable by humans and other computers
  - Reuse down to a fine-grained level
- Service Chains:
  - Assemble an application by connecting SOAP services
  - Portal “back end” should be all SOAP services
- Leverage Grid Workflow Engines:
  - SOAP Service Choreography

REST versus SOAP

- REpresentational State Transfer (REST)
  - Roy Fielding's Ph.D. Thesis
  - Everything is an object or document with a URI.
  - Problems:
    - Hidden state: cookies, sessions, CGI db, etc.
    - What about virtual datasets?
    - Need to generate too many URI's.
- Service Oriented Architecture (SOA)
  - Everything is a service.
  - What about stateful services? → WS-Resource
- Same Old Tradeoff
  - Like object-oriented programming versus functional programming (need both).
  - It's both an object and a service.

Amazon E-Commerce Services

- Item Search – SOAP Service input
  - <SubscriptionId>[Your SOAP key here]</>
    <Request>
      <Keywords>free lunch hoagie</>
      <SearchIndex>Books</>
    </Request>
- WSDL File
  - http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl
- Returns XML Document
  - Contains TotalResults count and a list of Items
  - Each item contains many attributes (title, authors, ASIN, etc.)

Amazon E-Commerce Services

- Also support REST paradigm for request
  - http://webservices.amazon.com/onca/xml?
    Service=AWSECommerceService
    &Operation=ItemSearch
    &SubscriptionId=[Your SOAP Key here]
    &Keywords=free%20lunch%20hoagie
    &SearchIndex=Books
    &ResponseGroup=Medium
- Response is same XML Document
  - With results tailored by ResponseGroup(s)
- Why are they opening their database up?
  - Allow 3rd parties to build apps on top of database
  - Build partnerships
  - Promulgate their software architecture/standard
  - Goal: Amazon lookup part of every business web app.
  - Makes sense even for commercial entities
  - Certainly best for scientific data archives

What is the SciFlo Network?

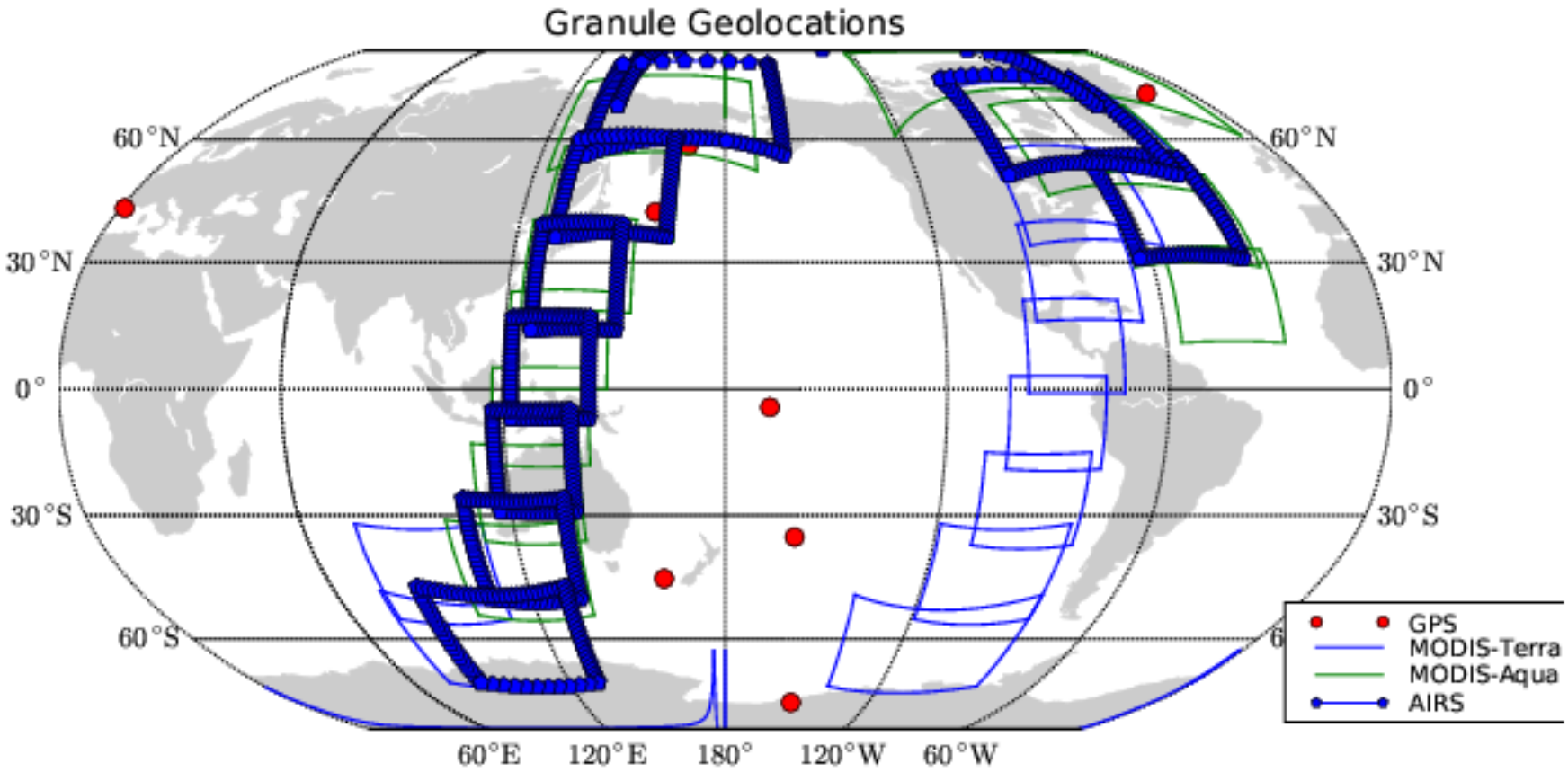
- "SciFlo" stands for Scientific Dataflow
- A Peer-to-Peer Network of Grid Workflow nodes
  - Author dataflow documents in XML
  - Tie together a series of operators into a complex dataflow: SOAP services, local executables or scripts, and python codes
  - Distributed dataflow execution engine
    - Choreographs parallel execution, can use many nodes
    - Data & operator movement is done by the engine
    - Each node serves data & operators, executes SciFlo documents, and is a client of other nodes.
- Leverages & reuses many XML technologies
  - Metadata described in XML and XML schema
  - Distributed computing via XML messaging (SOAP)
  - Service & operator interfaces described in XML (WSDL)
  - Ontologies & namespaces in XML (RDF & OWL)

SciFlo SOAP Services

- Time & Geolocation Querying for multiple datasets
  - GeoRegionQuery(timeRange, latLonRegion)
  - Same SOAP interface & web form for GPS, AIRS, MODIS, MISR
  - Query returns a list of unique object ID's (granules)
- Redirection/Caching Server for multiple datasets
  - GetDataById: Translates object ID's into list of on-line locations in DataPools or any SciFlo node (FTP or OpenDAP URL's)
  - DataPools & SciFlo P2P network are "crawled" to update redirection tables
  - No need to publish presence of data, continuously discovered
  - Distributed cache for scientific datasets: SciFlo engine automatically moves & caches datasets as required during dataflow executions
  - Can stage/cache input & generated datasets on local node

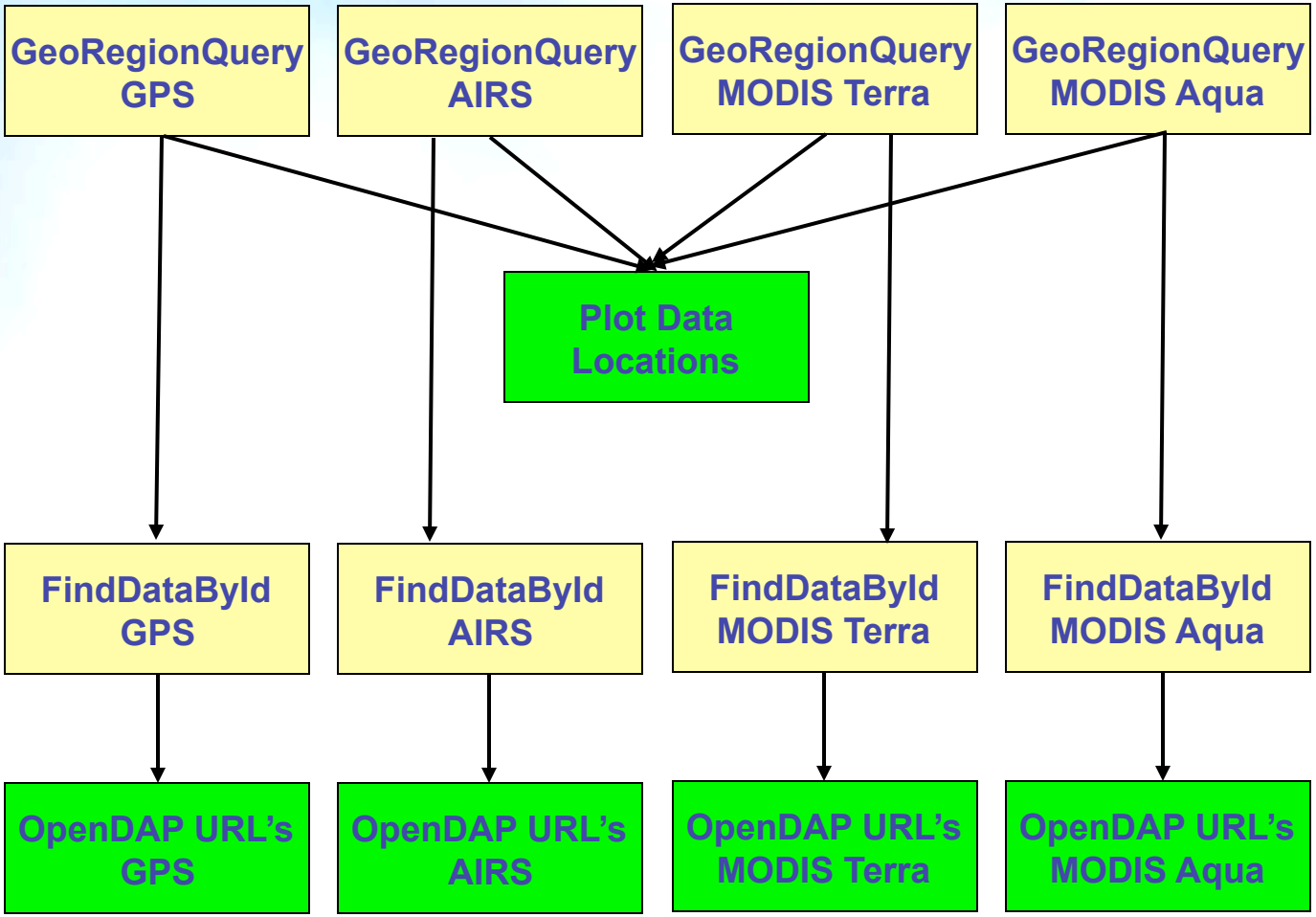
SciFlo: Scientific Knowledge Flow

Geolocation Plot for GPS, AIRS, MODIS



Try this SciFlo at http://sciflo.jpl.nasa.gov

Geolocation Query for GPS, AIRS, & MODIS



Provide Inputs & Execute the Flow

plotLocations_maplotlib.xml

Query AIRS, GPS, MODIS-Aqua, and MODIS-Terra for a specified GeoLocation, plot the locations, and provide urls to the data files.

Inputs

gpsVersion	1p0						
startDateTime	2003	/	01	/	03	00	: 00
endDateTime	2003	/	01	/	03	23	: 59
north	90.0						
south	-90.0						
west	-180.0						
east	180.0						

Outputs

locationMapFile: locs.png

OK

Example SciFlo Document

```
<?xml version = "1.0"?>
<sfl:sciflo xmlns:sfl ="http://genesis.jpl.nasa.gov/sciflo"
    xmlns:xsd ="http://www.w3.org/2001/XMLSchema">
  <sfl:flow name="plotLocations">
    <sfl:imports/>
    <sfl:description>Query AIRS, GPS, MODIS-Aqua, and MODIS-Terra for a specified GeoLocation,
                     plot the locations, and provide urls to the data files.</sfl:description>
    <sfl:inputs>
      <gpsVersion type="xsd:string">1p0</gpsVersion>
      <startDateTime type="xsd:ISODateTime">2003-01-03 00:00:00</startDateTime>
      <endDateTime type="xsd:ISODateTime">2003-01-03 23:59:59</endDateTime>
      <north type="xsd:float">90</north>
      <south type="xsd:float">-90</south>
      <west type="xsd:float">-180</west>
      <east type="xsd:float">180</east>
    </sfl:inputs>
    <sfl:outputs>
      <locationMapFile type="sfl:pngFile" processOutput="@plotLocations">locs.png</locationMapFile>
    </sfl:outputs>
    <sfl:processes>
      <sfl:process id="getAirsLocs" kind="sweet:geoRegionQuery">
        <sfl:inputs>
          <startDateTime/><endDateTime/>
          <north/><south/><west/><east/>
        </sfl:inputs>
        <sfl:outputs>
          <airsGranuleLocs type="sfl:xmlListOfLoc"/>
        </sfl:outputs>
        <sfl:operator type="soap">
          <sfl:description/>
          <sfl:binding>
            <sfl:wsdl>http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2AIRSData.wsdl</sfl:wsdl>
            <sfl:service>geoRegionQuery</sfl:service>
          </sfl:binding>
        </sfl:operator>
      </sfl:process>
```

Example SciFlo Document (continued)

```
      <sfl:process id="plotLocations">
        <sfl:inputs>
          <gpsMetadataXml from="@getGpsLocs"/>
          <modisTerraMetadataXml from="@getModisTerraLocs"/>
          <modisTerraMetadataXml from="@getModisAquaLocs"/>
          <airsMetadataXml from="@getAirsLocs"/>
        </sfl:inputs>
        <sfl:outputs>
          <plotFile/>
        </sfl:outputs>
        <sfl:operator type="method">
          <sfl:description></sfl:description>
          <sfl:binding type="pythonMethod">
            <sfl:name>plotLocs.plotLocations</sfl:name>
          </sfl:binding>
        </sfl:operator>
      </sfl:process>

      <sfl:process id="getAirsUrls" kind="sweet:getNativeGranuleById">
        <sfl:inputs>
          <metadataXml from="@getAirsLocs"/>
        </sfl:inputs>
        <sfl:outputs>
          <datasetLocs type="sfl:xmlListOfUrl"></datasetLocs>
        </sfl:outputs>
        <sfl:operator type="soap">
          <sfl:binding>
            <sfl:wsdl>http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2AIRSData.wsdl</sfl:wsdl>
            <sfl:service>getNativeGranuleById</sfl:service>
          </sfl:binding>
        </sfl:operator>
      </sfl:process>
    </sfl:processes>
  </sfl:flow>
</sfl:sciflo>
```

SciFlo: Scientific Knowledge Flow

Execution Status Page

you are at: home » Registered User's Area » MySciFlo **logged in as test** (log out)

SESSION id:**1118616183088532279 2**

gpsVersion:	**1p0**
startDateTime:	**2003/01/03 16:00:00 Universal**
endDateTime:	**2003/01/03 16:59:00 Universal**
north:	**90.0**
south:	**-90.0**
west:	**-180.0**
east:	**180.0**
locationMapFile:	**locs.png**

Work Unit Status/Color Legend: waiting, ready, working, done, and exception.

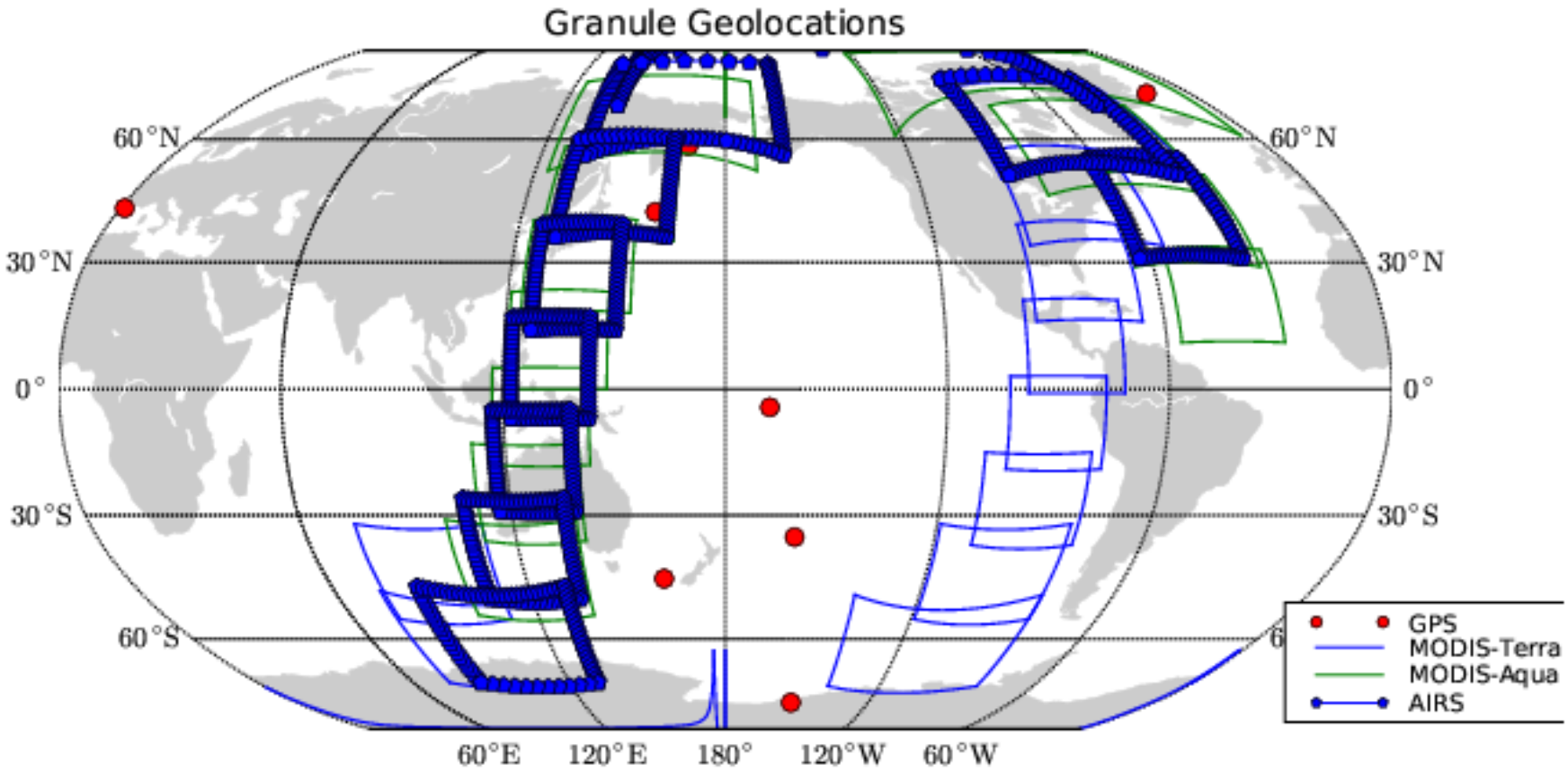
#	Call	Type	Dependencies	Status	Results	Exec Time(s)	Exception
1	L2AIRSData.wsdl:geoRegionQuery()	soap	[]	**done**	result	**7.52**	
2	L2GPSData.wsdl:geoRegionQuery()	soap	[]	**done**	result	**7.45**	
3	L2MODISTerraData.wsdl:geoRegionQuery()	soap	[]	**done**	result	**15.34**	
4	L2MODISAquaData.wsdl:geoRegionQuery()	soap	[]	**done**	result	**23.2**	
5	plotLocs.plotLocations	function	[2, 3, 4, 1]	**done**	result	**11.42**	
6	L2AIRSData.wsdl:getNativeGranuleById()	soap	[1]	**done**	result	**13.46**	
7	L2GPSData.wsdl:getNativeGranuleById()	soap	[2]	**done**	result	**13.33**	
8	L2MODISTerraData.wsdl:getNativeGranuleById()	soap	[3]	**done**	result	**6.41**	
9	L2MODISAquaData.wsdl:getNativeGranuleById()	soap	[4]	**done**	result	**0.21**	

Finished processing.

Outputs:

SciFlo: Scientific Knowledge Flow

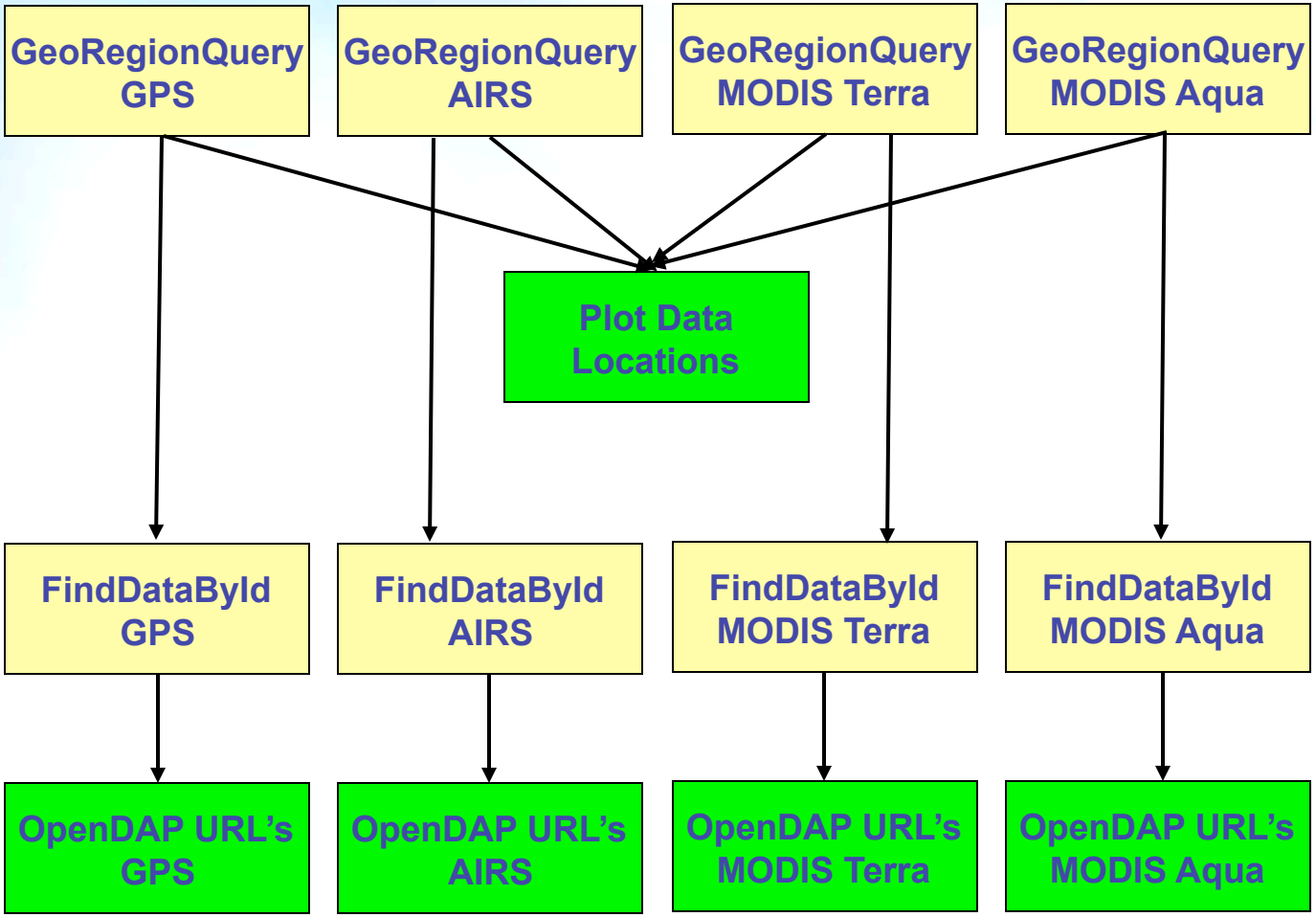
Geolocations Plot for GPS, AIRS, MODIS



Try this SciFlo at http://sciflo.jpl.nasa.gov

SciFlo: Scientific Knowledge Flow

Geolocation Query for GPS, AIRS, & MODIS



Technology Trends

- Pervasive use of XML – metadata & messaging
- Lightweight distributed computing – XML/SOAP messaging
- High performance binary data transfer – GridFTP or OpenDAP-g over optical fibers
- Google search – update index via crawler
- Scalable Peer-to-Peer (P2P) Networks
- Distributed queries & file sharing -- Napster
- Collaboration Networks – wiki, Friendster
- Semantic Web – meaningful ontologies in RDF & OWL

Technical Challenge

JPL

- Can one exploit all of these technologies and create a Grid Workflow system with the following features?
  - Lightweight, open source, user-installable (not root)
  - Runs on variety of hardware: Windows laptop to Linux cluster
  - P2P scalable network: goal of thousands of nodes
  - Each node generates & serves new, fused datasets
  - Distributed queries to discover newly generated data
  - Connect both remote services & local operators into a dataflow
  - Declare the dataflow in a simple XML document (not scripting)
  - Visual Programming (if desired, but not required)
  - Parallel execution with automatic load balancing, scheduling
  - Move algorithms/operators to the data
  - Publish & exchange scientific algorithms (dataflow docs.)
  - Preserves data provenance
  - Semantic annotations added to all products
  - Collaboration environment, shared web pages
  - Personal science notebook, personal data center

SciFlo is Multi-Purpose

JPL

- Each SciFlo client/server node is multi-functional:
  - Provides pre-configured SOAP services (e.g. GeoRegionQuery)
  - Serves data via an OpenDAP server and ftp or GridFTP
  - Provides a Redirection server: translate objectID -> file URL's
  - Contains metadata in a relational database (mysql)
  - Contains an XQuery-able XML document store (dbxml)
  - Executes SciFlo documents (dataflow execution engine)
  - Serves flow results on private & shared web pages (wiki)
- SciFlo Software Bundle
  - All Open Source, Push-Button Install on Linux & Windows
  - Installable by each user, root/admin privileges not required
  - One install provides pre-configured: SOAP services, OpenDAP server, redirection, ftp, mysql, dbxml, dataflow engine, & wiki.
- Personal Data Center for each scientist
  - Electronic scientific notebook (personal, configurable)
  - Collaborate by sharing wiki pages & exchanging SciFlo docs.

"The Grid" Concept

"On-Demand"
Virtualization

Evolving Grid Computing Standards

[From Globus Toolkit "Ecosystem" presentation at GGF11 by Lee Liming]

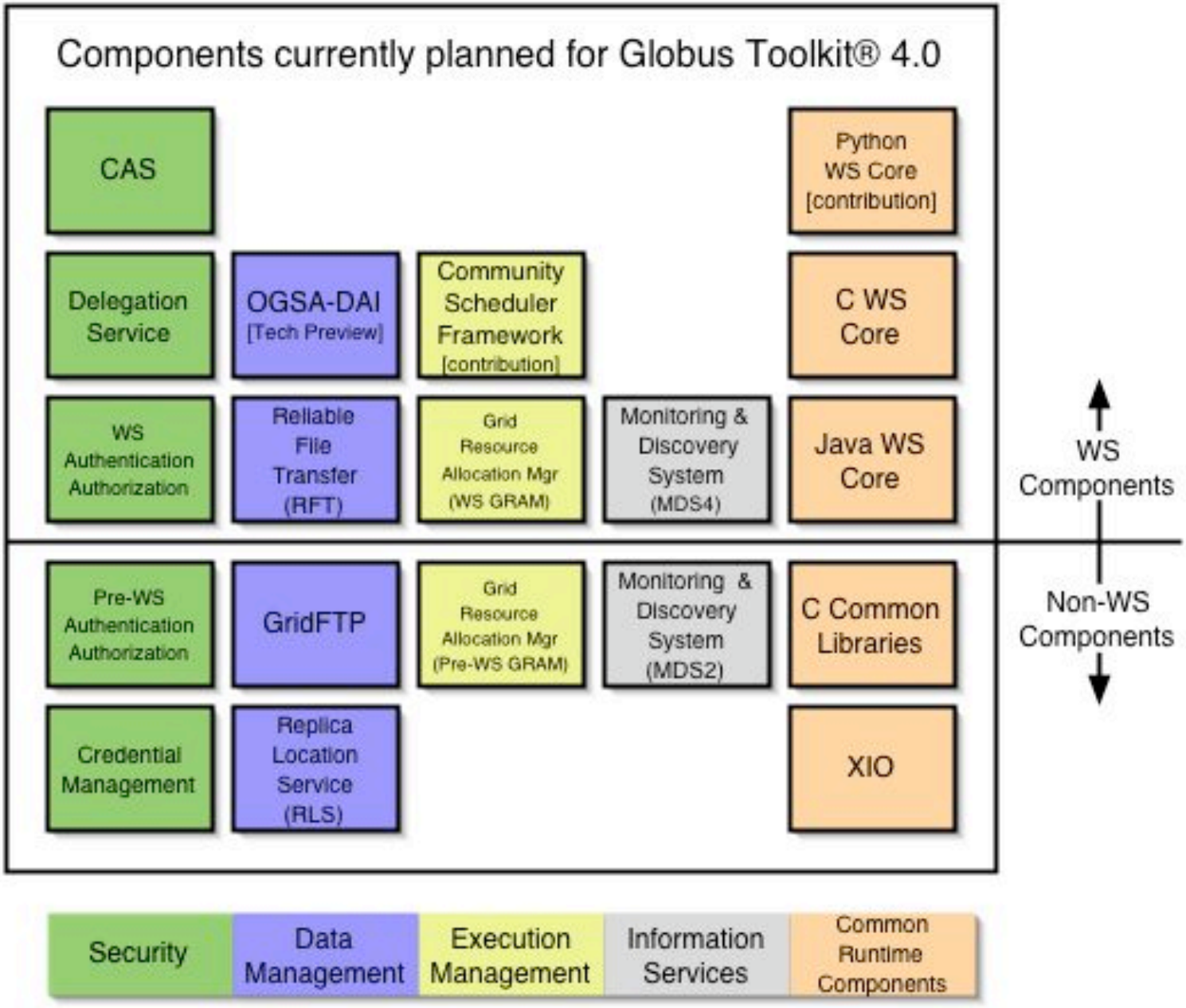
Evolving Grid Computing Standards

- History of Scientific Computing as a Utility
  - The Grid began as effort to tightly couple multiple super- or cluster computers together (e.g., Globus Toolkit v1 & v2).
  - Needed job scheduling, submission, monitoring, steering, etc.
  - SETI@HOME success
- OGSI: Open Grid Services Infrastructure
  - Globus v3.2 is open-source implementation using Java/C.
  - A service is Grid-enabled by inheriting from Java class.
  - Standard is complex and growing.
  - Challenge: Ease of installation & use.
- WS-Resource Framework (WSRF)
  - Capabilities treated as storage or computing resources exposed on the web – "stateful resources".
  - Globus Toolkit v4, May 2005

Evolving Grid Computing Standards

[From Globus Toolkit "Ecosystem" presentation at GGF11 by Lee Liming]

SciFlo: Scientific Knowledge Flow



Globus Components

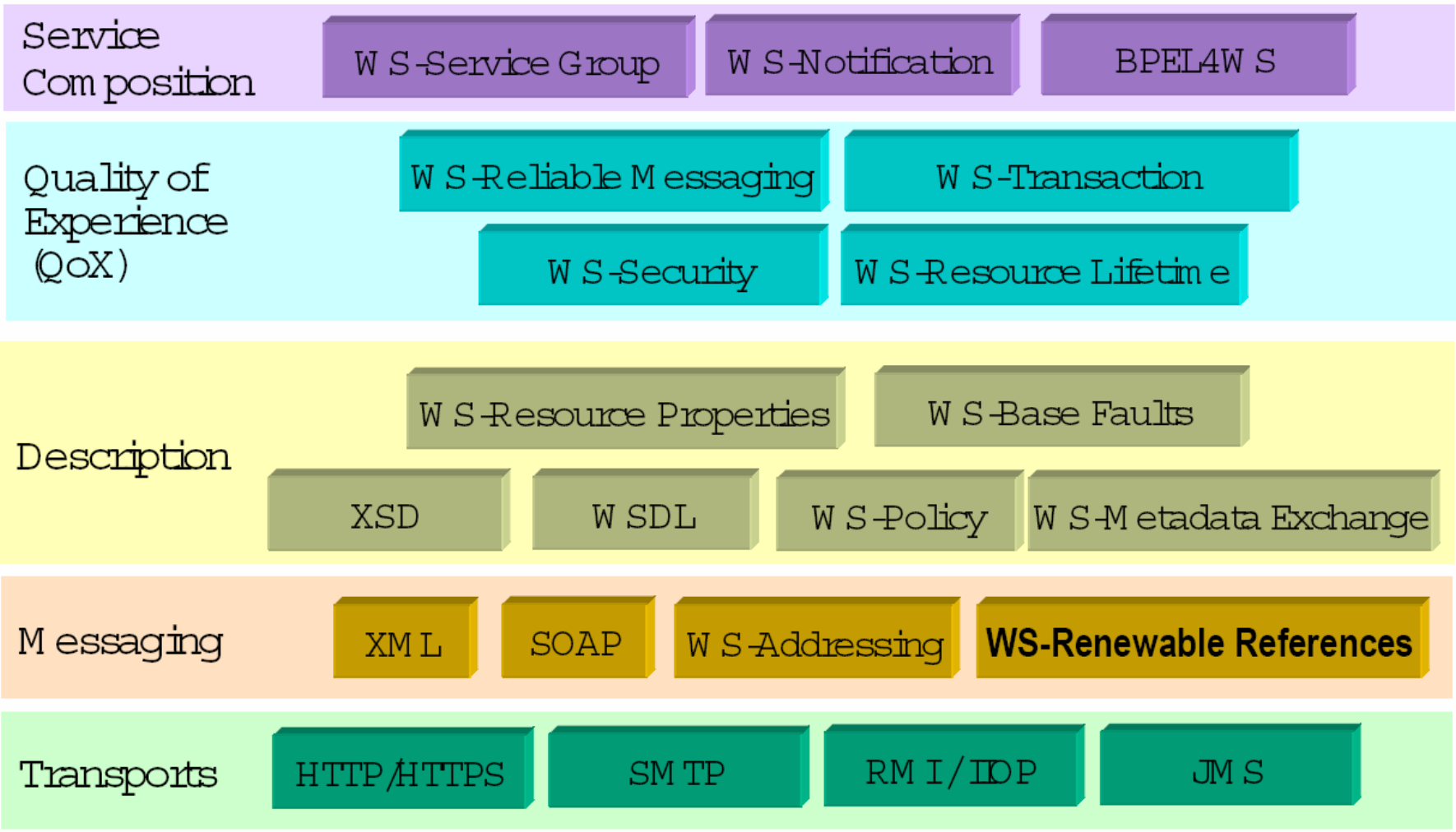
- *Security*
  - Grid Security Infrastructure (GSI)
  - Community Authorization Service (CAS)
  - MyProxy service (stores user credentials)
- *Monitoring & Discovery*
  - Monitoring and Discovery System (MDS)
  - Index Service (stores grid status information)
  - Trigger Service (resource change detection)
- *Grid Computing*
  - Globus Resource Allocation Manager (GRAM)
  - GRAM Scheduler Plug-ins
- *Data Grid*
  - Metadata Catalog Service (MCS)
  - Replica Location Service (RLS)
  - Reliable File Transfer (RFT)
  - GridFTP (multi-channel, fast ftp)

WS-* Alphabet Soup

Advanced Web Service Technologies

- WS-Security
- WS-Federation
- WS-Trust
- WS-Secure Conversation
- WS-Policy
- WS-Policy Attachments
- WS-Policy Assertions
- WS-Reliable Messaging
- WS-Addressing
- WS-BPEL
- WS-Atomic Transaction
- WS-Business Agreement
- WS-Coordination

WS Architecture



WS-Resource Framework Model

- **What is a WS-Resource**
  - **Examples of WS-Resources:**
    - **Physical entities** (e.g.. processor, communication link, disk drive) **or Logical construct** (e.g.. agreement, running task, subscription)
    - **Real or virtual**
    - **Static** (long-lived, pre-existing) or **Dynamic** (created and destroyed as needed)
    - **Simple (one), or Compound (collection)**

  - **Unique - Has a distinguishable identity and lifetime**
  - **Stateful - Maintains a specific state that can be materialized using XML**
  - **May be accessed through one or more Web Services**

SciFlo Services (I)

- Time & Geolocation Querying for multiple datasets
  - GeoRegionQuery(timeRange, latLonRegion)
  - Same SOAP & web form for GPS, AIRS, MODIS, & MISR
  - Query returns a list of unique object ID's (granules)
- Redirection/Caching Server for multiple datasets
  - FindDataById: Translates object ID's into list of on-line locations in DataPools or any SciFlo node (FTP or OpenDAP URL's)
  - DataPools & SciFlo P2P network are "crawled" to update redirection tables
  - No need to publish presence of data, continuously discovered
  - Distributed cache for scientific datasets: SciFlo engine automatically moves & caches datasets as required during dataflow executions
  - Can stage/cache input & generated datasets on local node

SciFlo Services (II)

- SciFlo: On-demand, distributed dataflow execution
  - executeFlow: Submit dataflow doc. for execution at any node
  - Move operators to data: Transfer executable or script to a SciFlo node "near" a large data archive
  - Move data to operators: Move individual variable grid via OpenDAP slicing or entire data file via ftp or GridFTP.
- Install custom operators in a SciFlo execution node
  - Custom subsetting and fusion operators pushed into the Langley and Goddard DAAC's (by end of 2005)
  - Optimize performance by distributing execution appropriately

SciFlo Data Query/Access (SOAP) Services

- GeoLocationQuery(startTime, endTime, lat, lon, timeTolerance, distanceTolerace, variable, metadataGroups)
  - Returns granule ID's and geolocation info. for EOS L2 swaths that intersect lat/lon point or are near enough.
- GeoRegionQuery(startTime, endTime, lat/lon region, . . .)
  - Returns granule ID's and geolocation info. for EOS L2 swaths that intersect the lat/lon region or are near enough.
- QueryByMetadata(variable, ListOfConstraints, groups)
  - Returns granule ID's and selected metadata for EOS granules that satisfy the metadata constraint expression: (min1 <= field1 <= max1 and/or min2 <= field2 <= max2 . . .).
- FindDataById(IdList, UrlOptions)
  - Given list of unique ID's, returns list of ftp, http, or DODS URL's pointing to the granules (files). Uses cache and redirection server.
- -- Other semantic interfaces possible

SciFlo Data Query/Access (SOAP) Services

- GetMetadata(type, groups)
  - Returns metadata describing scientific domain, dataset info, list of metadata fields, generic name translation table, location of XML schema documents, location of related ontologies (OWL/RDF), etc.
- GetHelp(type, groups)
  - A SOAP service that itself returns help documentation describing the available SOAP services.
- --These Services Combined Can Support:
  - Data Catalog, General Query, Data Access, Generic Names,
  - Data Discovery by Domain and Dataset Keywords
  - Type checking via XML schemas
  - Hooks to semantic web

Server Implementation

- P2P Server Infrastructure
  - Distributed catalogs of data sources and operator bundles.
  - Permanent hierarchical names for data & operators.
  - Data & operator movement handled automatically by server.
- SciFlo Execution Steps:
  - Validate: Parse & validate XML doc. against schema.
  - Queue: Push flow into an execution queue if slaves busy.
  - Embellish: Infer concrete workflow from abstract workflow; insert simple unit or format conversions, or more complex operators.
  - Plan/Schedule: Construct execution plan & annotate flow document accordingly.
  - Start Execution: Parallel execution using master & slave servers.
  - Forward: Forward partially evaluated flow to another server for load balancing & locality optimization.
  - Freeze/Thaw: Store execution state while waiting; wake up when long-running operators complete.
  - Deliver: Return URLs pointing to results (or fault) to SciFlo client, which then pulls output files (via http, sftp, or GridFTP).

Parallel, Asynchronous Dataflow Execution

- Parallelism at many levels during execution:
  - Create multiple processes on a single server node.
  - Create multiple threads within each process to execute, monitor, & respond to status queries.
  - Launch a sub-flow or operator on a slave node within the local computer cluster.
  - Invoke a remote operator via a SOAP or http CGI call and wait for results.
  - Redirect a flow or sub-flow to a server that can access a large data source locally.
  - Partially evaluate a flow and then redirect.
  - Launch a large, CPU-intensive operator on a Grid-enabled cluster or supercomputer (Sciflo and Grid interoperability).
  - Invoke the same flow multiple times if the source operators(s) yield multiple input objects (implicit file parallelism).
  - Switch execution between active flows while waiting for results from “frozen” flows.

Open Data Access Protocol

- www.opendap.org
  - Use a one-line query URL to retrieve a slice of a variable grid from a netCDF or HDF file anywhere in the world
  - Binary wire protocol for fine-grained data transfer
- OpenDAP URL
  - http://gen-dev.jpl.nasa.gov/genesis/cgi-bin/dods/nph-dods/genesis/data/airs/L2/20030113/airx2ret/AIRS.2003.01.13.171.L2.RetStd.hdf?TAirStd(1:3, 3:6, 4:17)
- OpenDAP Servers (via CGI)
  - netCDF, HDF, other file formats
  - Easy to implement another server
- OpenDAP clients
  - Matlab and IDL, any web browser
  - Python integration (Dominic Mazzoni)

Flexible Distributed Computing

- Move data to the operators – Move entire AIRS granules from the Goddard DAAC to a local SciFlo server at JPL. Compare AIRS to GPS locally.
- Move operators to the data – Move the comparison operators and the flow execution into a SciFlo server that has local access to the large AIRS granule archive.
- Custom parameter subsetting – Subset the AIRS granules at the DAAC using a custom subsetting operator to generate tailored granules. Transfer the smaller files to a local SciFlo server.
- On-demand OpenDAP Slicing – Leave the AIRS granules in the Goddard Data Pool and perform the comparison at JPL. Using OpenDAP, only grab the subset of variables (temp, water vapor, etc.) that are needed for the comparison.

Thinking in XML

- Design the XML metadata and data representations first. Add an ontology.
  - Then create objects in your favorite programming language.
  - Then create objects in more languages for broader use.
- 4DRectilinearGrid datatype
  - <4DRectilinearGrid>
    <Latitudes>
    <Series>-90, 90, 2
    <Longitudes>@http://gsfc.nasa.gov/cgi-bin/dods/AIRS.hdf?Longitude
    <Altitudes>1, 5, 10, 50, 100, 500, 1000
    <Times>
    <Variable>
    <Name>
    <Unit>
    <MissingValueMarker>
    </Variable>
    <Grid>@http://gsfc.nasa.gov/cgi-bin/dods/AIRSgranule.hdf?TAirStd
    </Grid>
    </4DRectilinearGrid>

Hybrid XML/Binary Serialization

- Composite Object structure
  - Serialize to XML
  - Load into python, or other languages
- Binary Data vector, matrices, grids
  - Save in netCDF or HDF files
  - Or refer to remote OpenDAP slices
- "Smart" netCDF files
  - File contains usual data variables (matrices)
  - Also contains URL's pointing to composite object structure serialized in XML
  - Example: HDFEOS conventions enforced in object structure

New Paradigm for Scientific Computing

- Use two or more channels
  - XML Messaging on one
  - Binary data protocol (OpenDAP, GridFtp) on the others
- Two channels can use different transports
  - XML Messaging over HTTP or dark P2P
  - OpenDAP over LambdaRail or scalable P2P caching network
- XML Messaging Channel for:
  - Control, Service Chaining, Workflow, Metadata Exchange
  - Configuration of Binary channel
- Binary Data Channel(s)
  - Out-of-band from XML point of view
  - Bulk data transfer, replication, or caching
  - Could be multi-protocol (OpenDAP, GridFtp, LambdaRail)

Why Use SciFlo?

- New paradigm of peer-to-peer scientific collaboration
  - Each P2P SciFlo nodes generates & serves fused datasets
  - Exchange SciFlo docs & operators to share analysis algorithms
  - Dataflow results appear as new web pages in a shared wiki
  - Collaborate by sharing wiki pages (groups read & annotate)
  - Personal data center & analysis notebook for each scientist
- P2P workflow is a scalable solution for large-scale data fusion.
- Author SOAP Services without knowing anything about SOAP or WSDL.

SciFlo As an Authoring Toolkit

- Assemble operators by writing XML document
  - Connect SOAP data query/access services to custom, executable algorithms written by scientists
  - IDL, MATLAB, or python codes can become operators
- SciFlo Engine automatically:
  - Generates web (HTML) form to call the flow
  - Publishes custom flow as a new SOAP service (if desired)
- Create many SOAP Services Automatically
  - No glue or SOAP code to write
  - Only write scientific algorithms, in language of choice
  - New services have semantic web annotations
- Publish Analysis Flows
  - Exchange SciFlo documents
  - Generated products have lineage & semantic annotations

Serve Generated Datasets

- **Domain**: Multi-Instrument Atmospheric Science
- **Variables**: temperature, water vapor, clouds, aerosols
- GENESIS GPS limb scan database
  - Temperature & water vapor profiles, 1999-present
- Staged AIRS, MODIS, and MISR L2 & L3 granules
  - Temp, H_2O, cloud properties
  - Selected periods, 2003-present, and redirect to DataPools
- Merged, co-registered atmospheric datasets (netCDF)
  - Example: AIRS & GPS temp/H_2O matchups, Jan. 2003-
  - Example: AIRS & MODIS cloud matchups
- Multi-instrument analysis results
  - Example: Statistics & plots for AIRS temp & H2O versus GPS
- Virtual Datasets: On-demand analysis by executing SciFlo dataflow docs.

Implementation Challenges

- Can we converge to a core set of services?
  - Often each SOAP service has slightly different semantics
  - Need to agree on core semantics
    - Start with time & geolocation querying
  - Can automatic semantic mediation solve the problem?
- Interoperability with existing services
  - Redirect GeoRegionQuery to ECHO query service or a Web Coverage Server (WCS)
  - Two channel system: XML messaging & binary OpenDAP
- Interface: REST versus SOAP
  - REST: REpresentational State Transfer: http GET URL's
  - SOAP: http POST of a hierarchical XML doc.
  - No need for a religious war
  - Can flatten XML doc. into one-line query URL
    - See Amazon E-Commerce web services

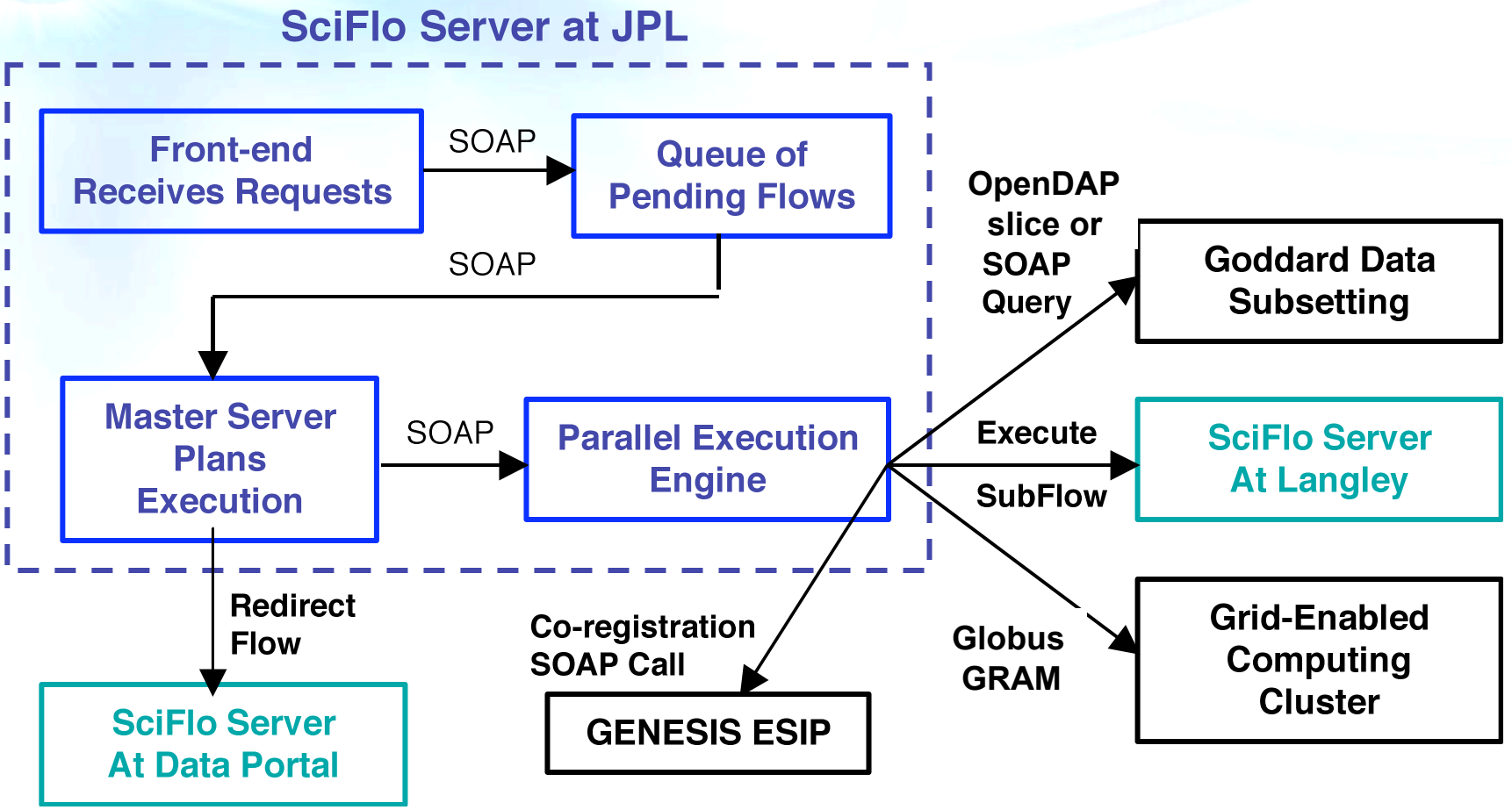
Benefits of Using Web Services

- Pervasiveness of SOAP/WSDL/UDDI in business
  - Lightweight, loosely-coupled distributed computing
  - SOAP request/response are a few lines of code in all languages
  - Many free & commercial tools for dev, monitoring, etc.
  - Services are an easy way to reuse code & algorithms
  - SOAP can use many transport protocols (some asynchronous)
    - Email (smtp), Reliable Messaging (JMS, IM)
    - SOAP routing at the hardware switch level
- Grid Standards / Grid Workflow
  - Exploit WS-* standards: -Security, -ResourceFramework
  - The Grid: Provision compute & storage resources on demand
  - Numerous Workflow engines: SciFlo, BPEL, Kepler
  - Reuse: Services are reused algorithms, SciFlo doc. exchange
  - P2P Collaboration: Hundreds of interacting SciFlo nodes
  - SciFlo Empowerment: Personal data center for everyone!

Barriers to Infusion

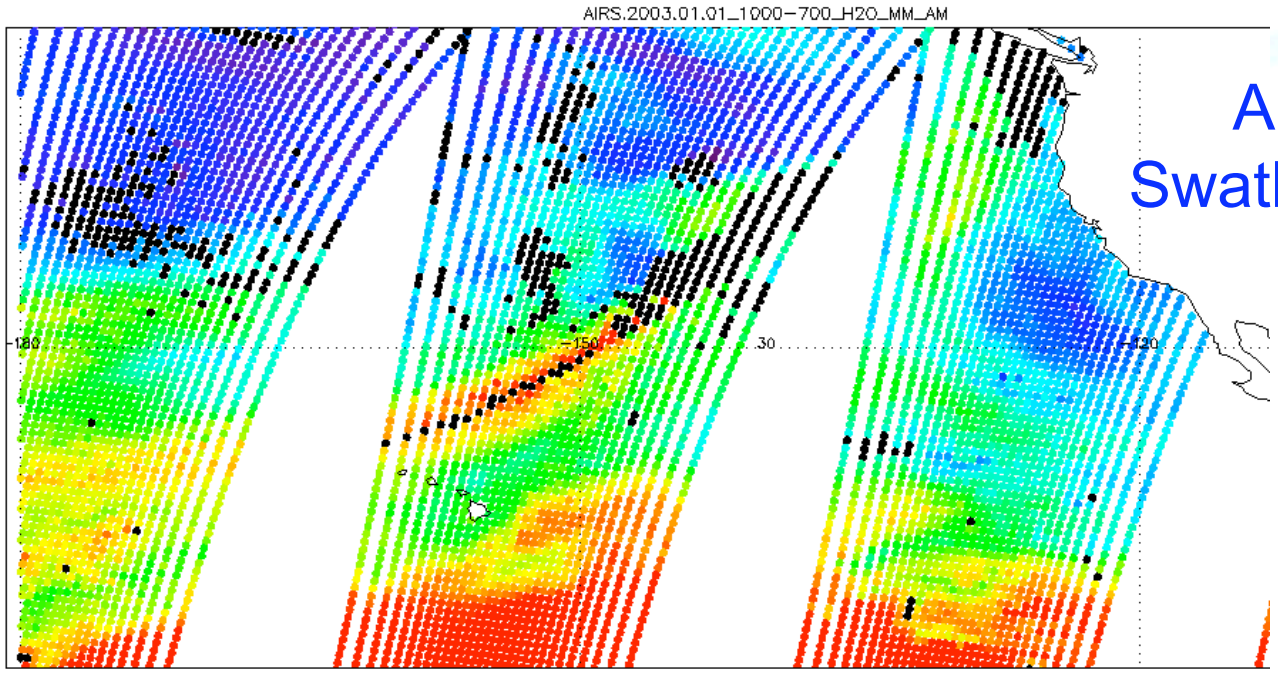
- Technology Adoption
  - Yes, creating structured XML is work. But yields great benefits.
  - SOAP is hard: Balderdash, only in Fortran.
  - Religious Debates: REST vs. SOAP
    - Is it a service or an object? It's both. (Tastes great, less filling.)
- Interface Standards: Whose semantics?
  - Interoperate with & reuse WMS/WCS and OpenDAP
  - Semantic mediation: e.g., rewrite XML queries
  - P2P: Only two need to agree to collaborate & fuse.
- Sociological
  - I'll do it all myself. I don't trust your algorithms / services.
    - Can't do everything
    - Can publish services & fusion (dataflow) algorithms.
  - Open source: low cost & transparency drives adoption

Distributed Computing Using SciFlo



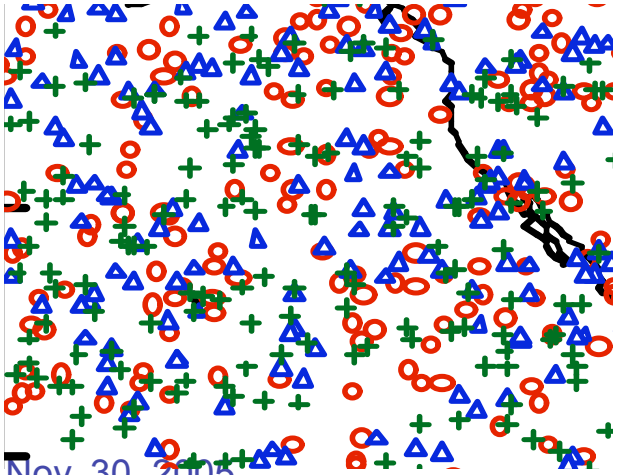
Inject data query or flow execution request into SciFlo network from any node.

AIRS/GPS Co-registration: Point to Swath



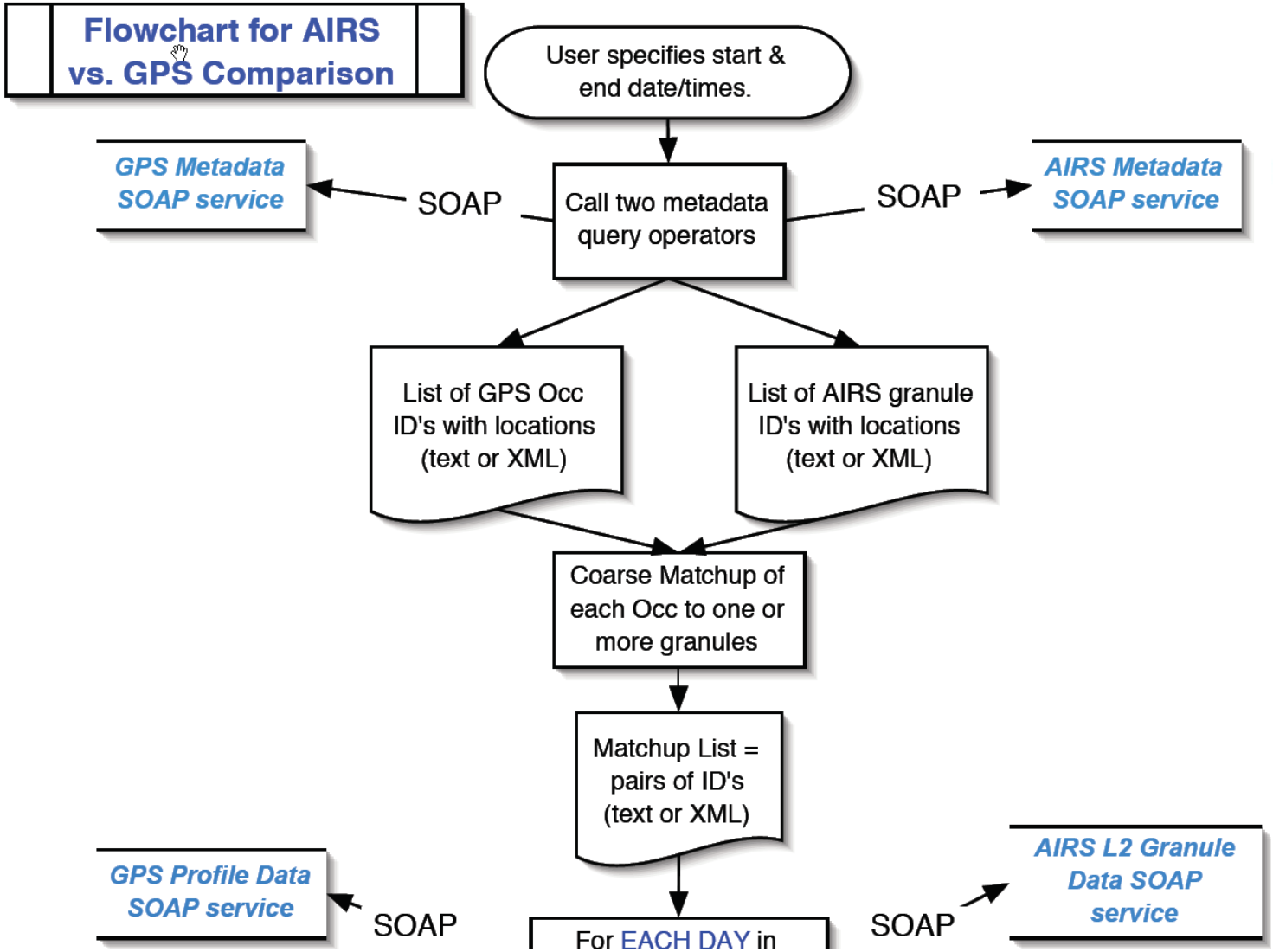
AIRS Level2
Swaths over Pacific

GPS Level2
Profile Locations



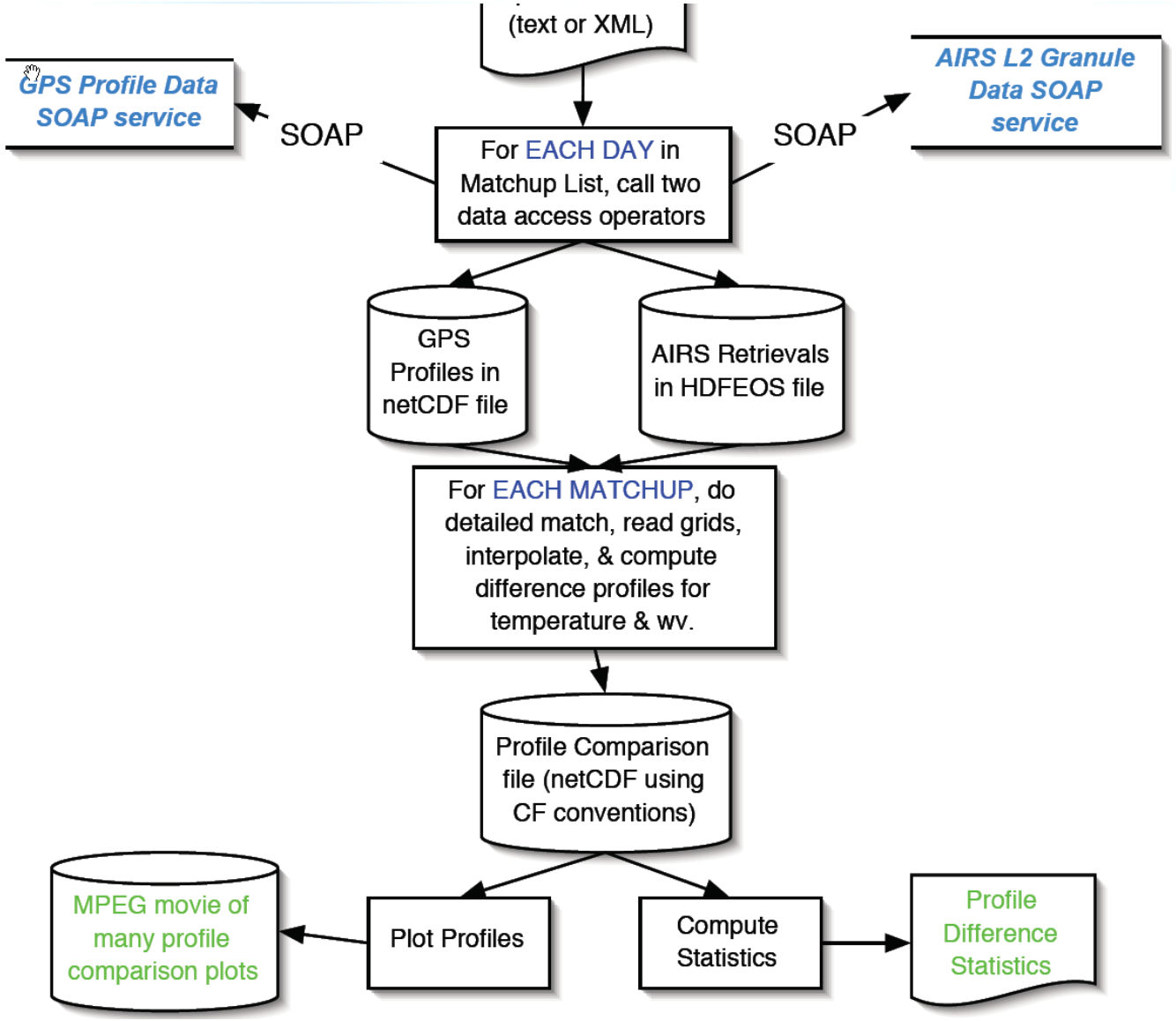
SciFlo: Scientific Knowledge Flow

AIRS versus GPS Flowchart (part 1)



SciFlo: Scientific Knowledge Flow

AIRS versus GPS Flowchart (part 2)



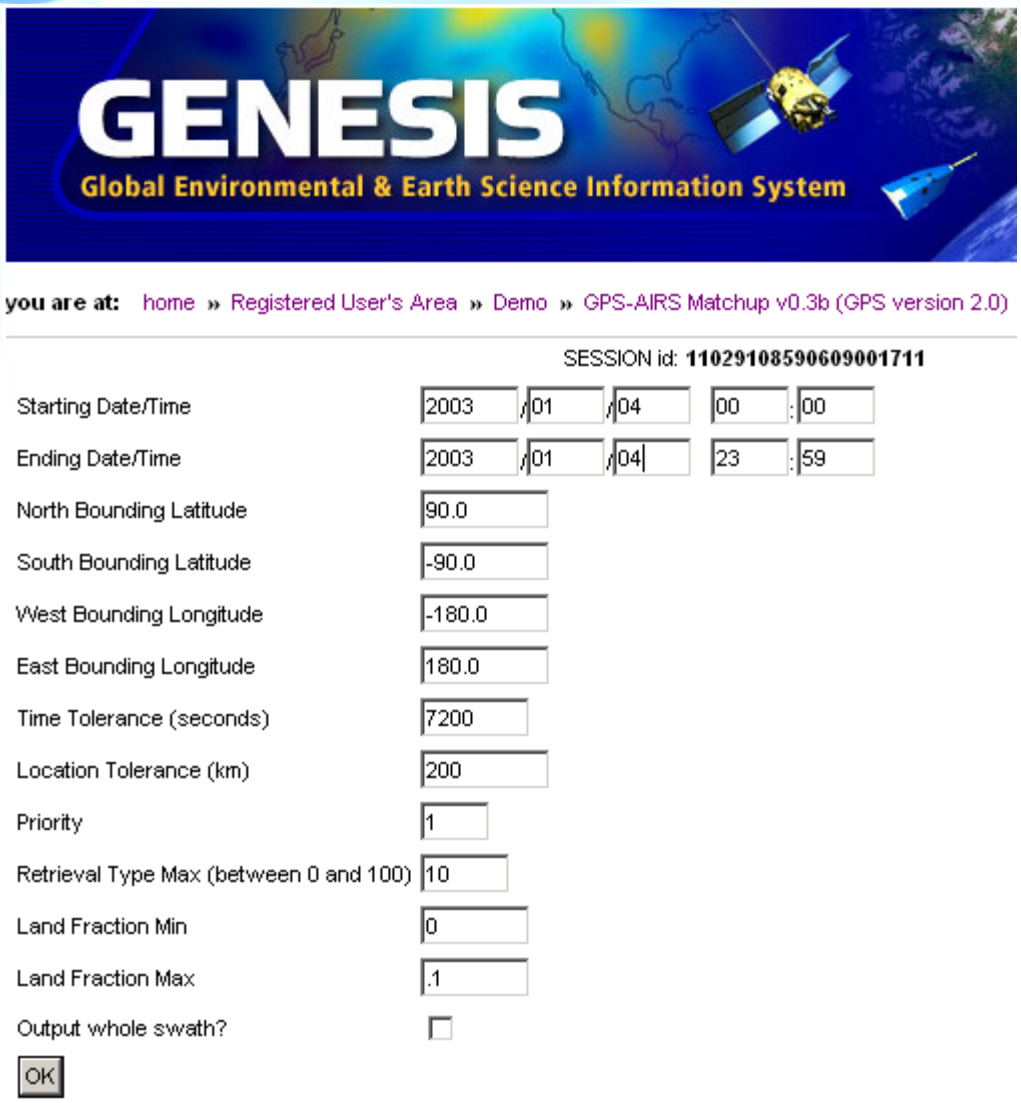
SciFlo: Scientific Knowledge Flow

SciFlo Document

```
<?xml version = '1.0'?>
<sciflo xmlns:sfl = 'http://genesis.jpl.nasa.gov/sciflo'
    xmlns:xsd = 'http://www.w3.org/2001/XMLSchema'>
  <sfl:flow id='gpsAirsCmp'>
    <sfl:imports>
    <sfl:description>
Match up GPS occultation profiles with AIRS L2 temperature and water vapor retrievals
and generate comparison plots and difference statistics.</sfl:description>
    <sfl:inputs>
      <startDateTime type='xsd:ISODateTime' kind='sfl:startTime'>2003-01-06T00:00:00</startDateTime>
      <endDateTime type='xsd:ISODateTime'>2003-01-06T23:59:59</endDateTime>
      <timeMatchTolerance type='xsd:int' unit="seconds">600</timeMatchTolerance>
      <distanceMatchTolerance type='xsd:float' unit="km">1000.0</distanceMatchTolerance>
      <priority>1</priority>
      <retrievalTypeMax type='xsd:int'>10</retrievalTypeMax>
      <landFractionMin>0.0</landFractionMin>
      <landFractionMax>0.1</landFractionMax>
      <outputWholeSwath type='xsd:boolean'>False</outputWholeSwath>
      <gpsFileName type='xsd:string' defaultValue='gps.nc'>gps.nc</gpsFileName>
      <matchupFileName type='xsd:string' defaultValue='match.nc'>matchup.nc</matchupFileName>
      <plotFileName defaultValue='plot.tar.gz'>plot.tar.gz</plotFileName>
    </sfl:inputs>
    <sfl:outputs>
    <sfl:processes>
      <sfl:process id='getAirsLocs' kind='sfl:GetIdAndLocationMetadata'>
        <sfl:inputs>
          <startDateTime>@inputs.startDateTime</startDateTime>
          <endDateTime>@inputs.endDateTime</endDateTime>
        </sfl:inputs>
        <sfl:outputs>
          <airsGranuleLocs type='sfl:xmlListOfLoc'/>
        </sfl:outputs>
        <sfl:operator>
          <sfl:description></sfl:description>
          <sfl:binding>
            <sfl:module name='GenesisService' type='soapService'>
              <sfl:url>http://gen-dev.jpl.nasa.gov/genesis/GenesisService.wsdl</sfl:url>
            </sfl:module>
            <sfl:method>getAirsMetadataByDateRange</sfl:method>
          </sfl:binding>
        </sfl:operator>
      </sfl:process>
```

SciFlo: Scientific Knowledge Flow

AIRS & GPS Retrievals Matchup Demo



- **Interface:** HTML web form auto-generated from XML dataflow doc.
- **Input:** User enters start/end time & other co-registration criteria.
- **Flow Execution:** Calls 4 SOAP data query services & total of 15 operators on 4 computers.

SciFlo: Scientific Knowledge Flow

AIRS & GPS Retrievals Matchup Demo

Work Unit Status Color Legend: waiting, ready, working, done, and exception.

Work Unit #	Call	Type	Dependencies on other Work Units	Status	Pertinent Results	Execution Time(s)	Exception
1	http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2AIRSData.wsdl: findByTimeAndLocationBox()	soap	None	done		1.94	
2	utils.getGranuleidCountFromXml	function	1	done	Got 455 AIRS granuleids	0.15	
3	http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2AIRSData.wsdl: getMetadataByIdList()	soap	1	done		3.53	
4	utils.getAirsMetadataDictFromXml	function	3	done		0.32	
5	http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2GPSData.wsdl: findByTimeAndLocationBox2p0()	soap	None	done		1.35	
6	utils.getDatasetidCountFromXml	function	5	done	Got 159 GPS datasetids	0.05	
7	utils.getGpsMetadataDictFromXml	function	5	done		0.11	
8	gps_airs_match2.coarseMatchup	function	(4, 7)	done	Got 61 GPS-AIRS coarse matches	0.15	
9	utils.getSortedGpsDatasetidsXml	function	8	done		0.01	
10	http://gen-dev.jpl.nasa.gov:8080/genesis/wsdl/L2GPSData.wsdl: aggregateOverIdList2p0()	soap	9	done		11.71	
11	utils.getGpsNetcdf	function	10	done	download GPS generic grid NetCDF	0.16	
12	utils.getLocalAirsGranuleFiles	function	8	done		0.0	
13	utils.downloadAirsFiles	function	12	done	downloaded 0 of 0 AIRS files	0.07	
14	gps_airs_match2.findGpsScanMatch	function	(8, 11, 12, 13)	done	Got 26 GPS-AIRS scan matches; download matchup NetCDF	2.65	
15	utils.createSwathTarball	function	12	done	download tarball of matching AIRS swaths	2.13	
16	gps_airs_plot.plot	function	14	done		12.28	
17	utils.createPlotsTarball	function	16	done	download tarball of matchup plots	23.89	
18	utils.makeFlashMovie	function	16	done	download flash movie of matchup plots	23.27	

Finished processing.

Sum of individual execution times of work units: 83.77 seconds.

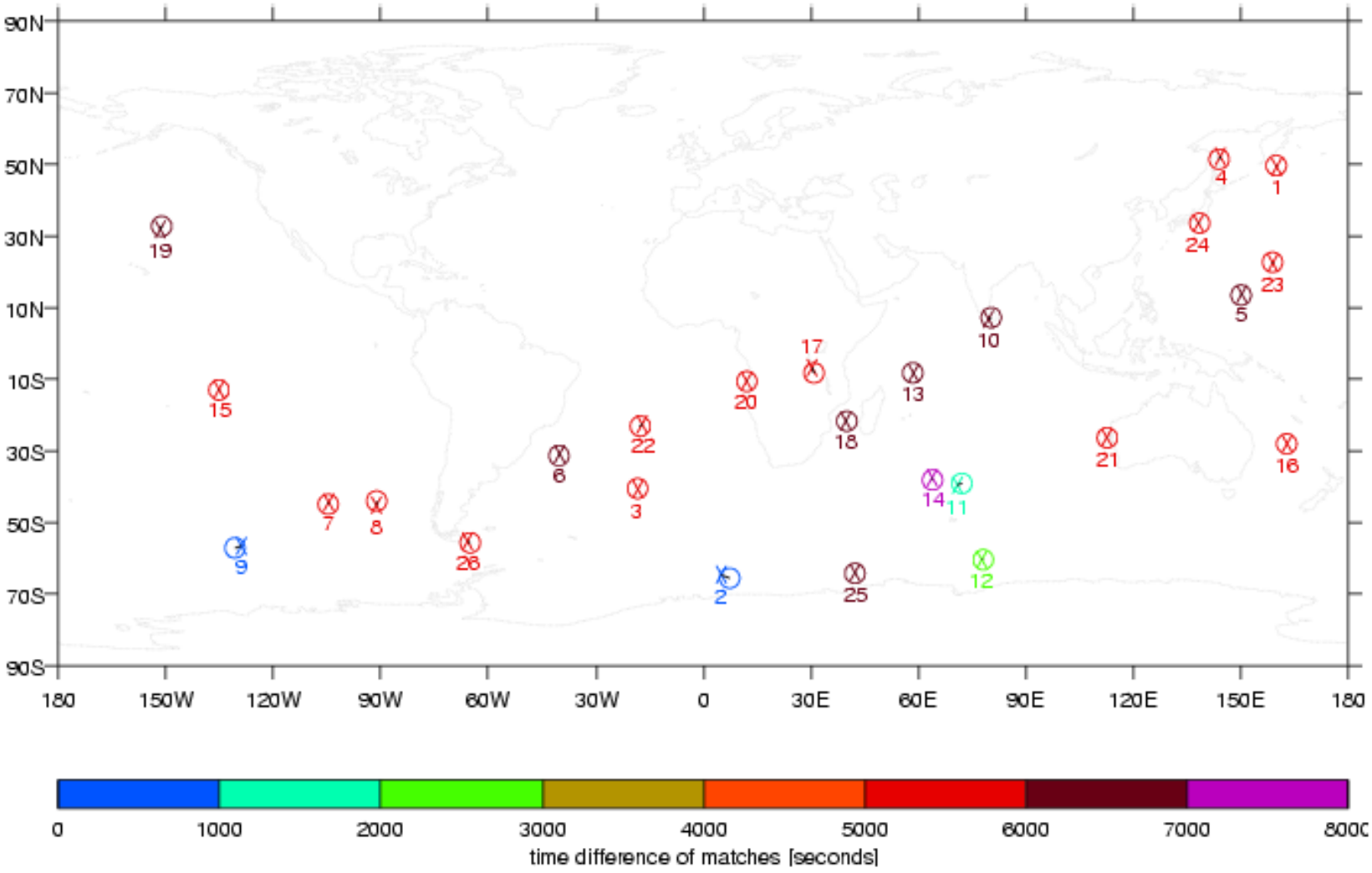
Total time spent executing work units in parallel: 80.71 seconds.

Flash movie:

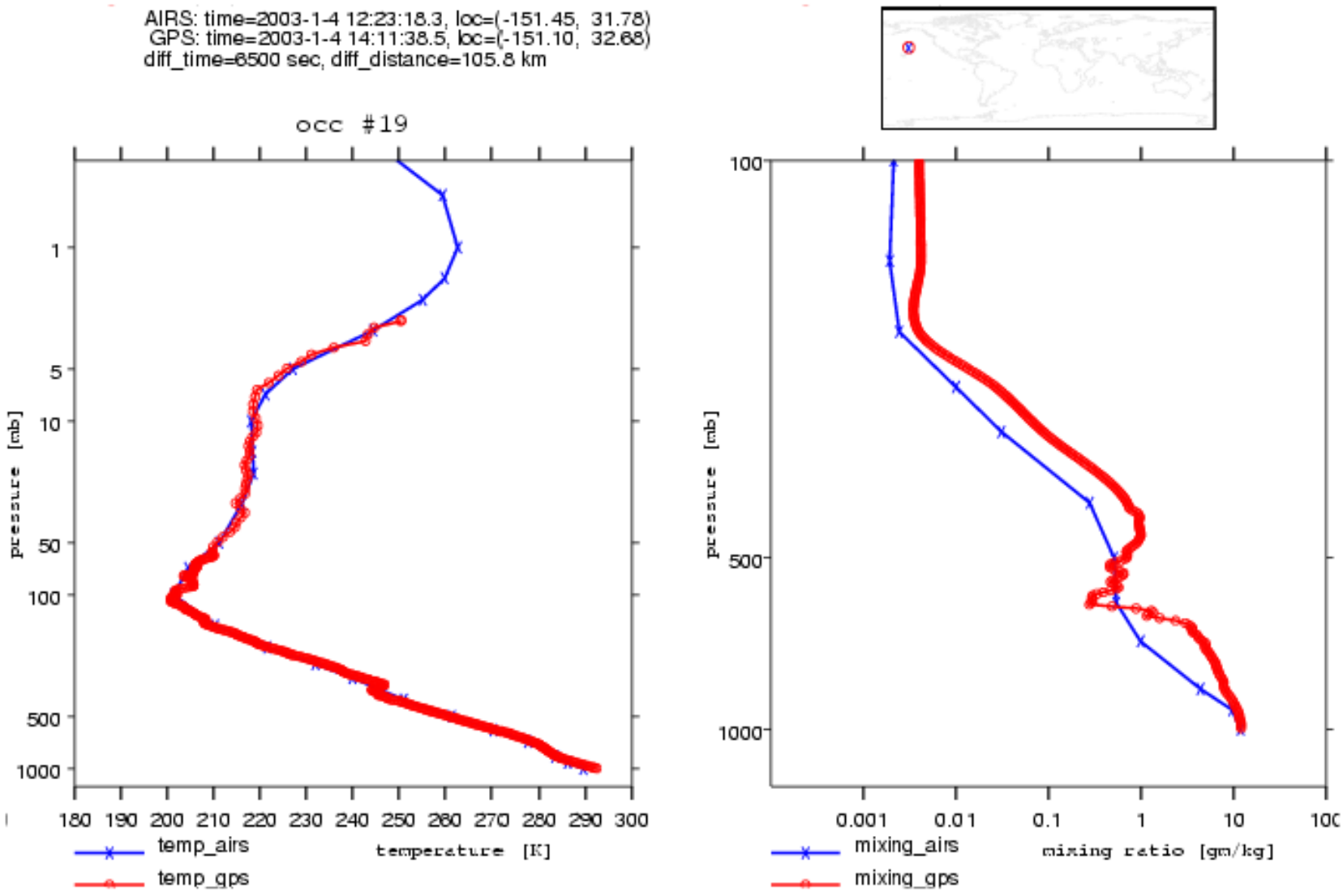
- **Results Page:** Shows status updates during execution and then final results.
- **Caching:** Reuse intermediate data products or force recompute.
- **Results:** Merged data in netCDF file & plots as Flash movie.

SciFlo: Scientific Knowledge Flow

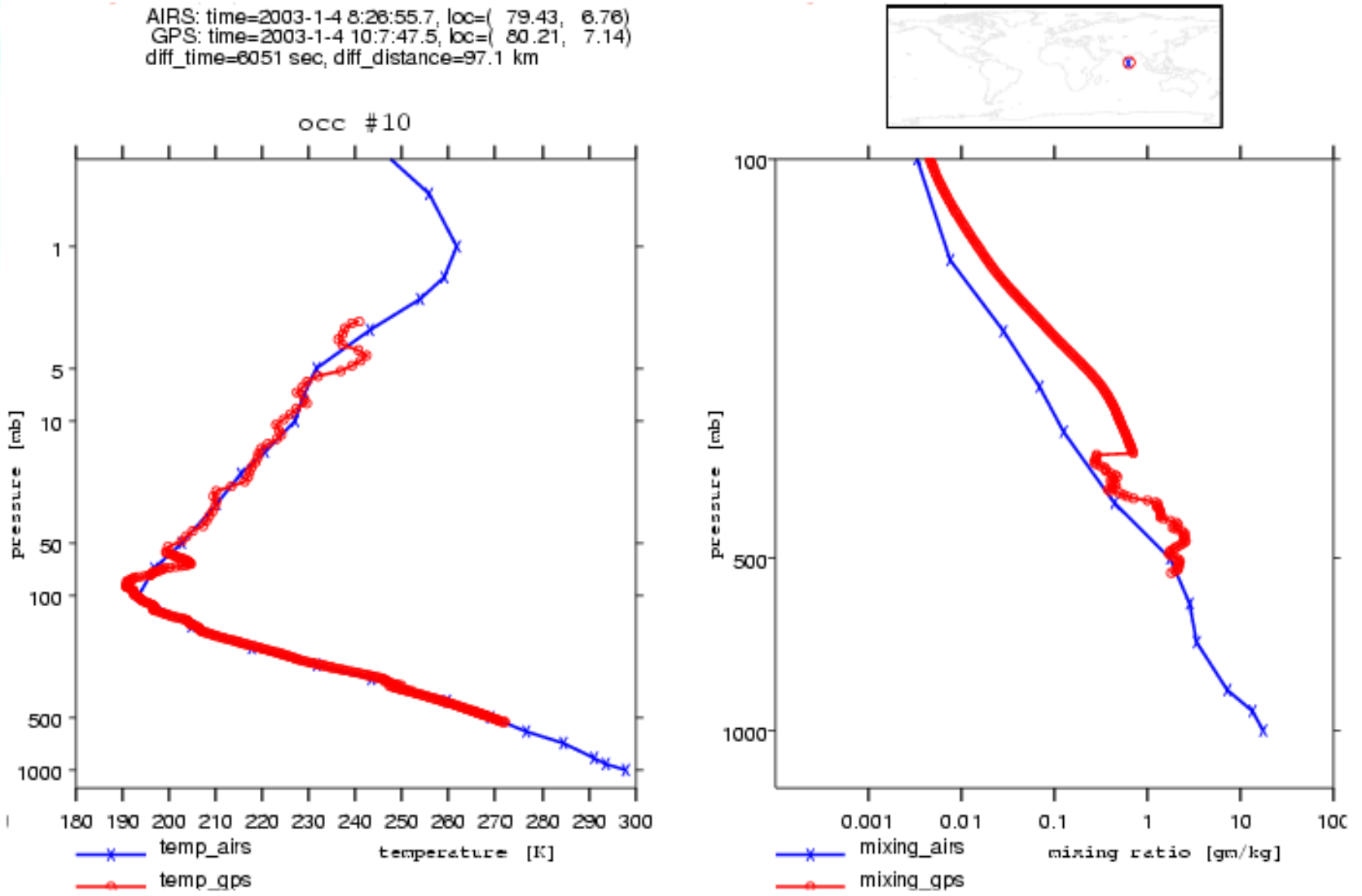
AIRS/GPS Matchups



AIRS/GPS Temperature & Water Vapor Comparison Plots



AIRS/GPS Temperature & Water Vapor Comparison Plots



Technology Summary

- SciFlo is a deep integration of multiple IT's:
  - XML standards: schema, SOAP, WSDL, UDDI, RDF, OWL
  - Distributed computing via XML messaging
  - Fine-grained, on-demand, binary data transfer via OpenDAP
  - Metadata in queryable XML and relational databases
  - Grid Workflow: WS-ResourceFramework, WS-* standards
  - The Grid: Compute & storage resources as a utility
- Not Six, But Thousands of Nodes (P2P)
  - Distributed Systems are Replacing Centralized Systems
  - New Paradigm: Lightweight, Loosely-Coupled, Distributed
  - Challenge: Globus toolkit is not lightweight.
  - For now, use only pieces of Globus: Security (GSI), GridFTP
  - Add more from WS version of Globus (v4) as they mature

Future Plans

- Version 1 of SciFlo available Fall 2005
  - Deploy within major data centers soon
  - Earth science: Goddard, Langley, JPL DAAC's, AIRS team
  - Aerosol virtual observatory (AMAPS ACCESS grant, Amy Braverman, MISR team)
- Apply SciFlo to planetary and space science data
  - Bridge to OODT, workflow for Planetary Data System
  - Large-scale Grid workflow as a LambdaRail "killer app"
- Adopt Grid technologies as they mature (Globus v4)
  - GridFTP, WS-Security/GSI, GRAM job submission
- Add more Semantic Web Capabilities
  - Ontology for operator types and geo-located data (SWEET)
- World Domination: Thousands of Nodes

Summary

- SciFlo's Innovation Lies in Combining Many Elements into a Single Open-Source System
  - Abstract XML dataflow documents
  - Semantic inference using XML metadata
  - Parallel dataflow execution engine
  - Move operators to the data.
  - Every node is both client & server; easy node replication.
  - SOAP architecture, but also P2P functionality.
  - Initiate grid computations from your desktop.
- P2P Network:
  - SciFlo nodes inside all Science Data Centers
  - Personal science notebook for each scientist
- Multi-Instrument Earth Science
  - Multi-Instrument Science Portals
  - Large-scale multivariate statistical studies and verification of weather/climate models.